Universidad Autónoma de Querétaro
Facultad de Informática
Especialidad En Sistemas de Información

Algoritmo automático de detección y aforo vehicular en tiempo real en horario diurno.

Opción de titulación
Tesis

Que como parte de los requisitos para obtener el Grado de
Maestro en Sistemas de Información: Gestión y Tecnología

Presenta:
Lic. José Alejandro Ascencio Laguna

Dirigido por:
Dra. Sandra Luz Canchola Magdaleno

Dra. Sandra Luz Canchola Magdaleno. _____ Firma
Presidente

Dra. Ma. Teresa García Ramírez. _____ Firma
Secretaria

M.I.S.D. Carlos Olmos Trejo. _____ Firma
Vocal

M.S.I. Fausto Abraham Jaques García. _____ Firma
Suplente

M. en C. Verónica Josefina Soria Anguiano. _____ Firma
Suplente

M. en C. Ruth Angelica Rico Hernández
Director de la Facultad de Informática

Dr. Irineo Torres Pacheco
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Octubre 2014

RESUMEN

En este trabajo se ha implementado una arquitectura para el conteo y clasificación vehicular. Se aplica la Computación Acumulativa para detectar al conjunto de vehículos en movimiento y discriminar los píxeles estáticos, se hace la segmentación a través de Funciones de Densidad de Probabilidad (FDP) de Ventanas de Parzen, para obtener un núcleo Gaussiano en cada objeto; la altura de cada pico de Gauss representa el volumen del vehículo. Por último se hace el análisis de varianza ANOVA, para validar la similitud de los vehículos y seguirlos de un instante al otro. Se obtuvo un conjunto de muestras para determinar los umbrales de calibración en la eliminación de ruido y píxeles estáticos, anchos de ventana para el cálculo de las FDP, el nivel de significancia en los análisis de similitud y la eficiencia en el algoritmo de seguimiento propuesto. Los resultados obtenidos pueden ser mejorados, ya que al incrementar el número de objetos en escena, se alcanzan niveles de error del 54%, los métodos evaluados por separado dan buenos resultados. La detección de objetos junto con el análisis de similitud y una buena calibración, puede llegar a niveles de eficiencia del 95%. Los altos niveles de error en la evaluación del sistema se deben al lugar experimental, ya que se intentó cubrir con una sola cámara los ocho accesos de la escena, y esto provoca un gran número de objetos ocluidos. Se propone como trabajos futuros implementar un sensor de video en cada acceso y utilizar la técnica de ANOVA para analizar la similitud entre los objetos detectados en una cámara y otra, o cambiar la experimentación a una carretera de un solo carril.

(**Palabras clave**: Computación Acumulativa, Ventanas de Parzen, ANOVA)

SUMMARY

In this paper has been implemented an architecture for counting and vehicle classification. Accumulative Computation is applied to detected the set of moving vehicles and discriminate the static pixels, segmentation is done by Probability Density Functions (PDF) of Parzen windows to obtain a Gaussian Kernel on each object; the height of each Gauss peak represented the vehicle dimension. Finally is done the ANOVA Variance Analysis to validate the similitude of vehicles and tracking them from one instant to another. A set of samples were obtained to determinate the thresholds calibration on the denoising and static pixels, window widths to the calculation of the PDF, the significance level on the similitude analysis, and the efficiency in the tracking algorithm proposed. Results obtained can be improved, because by increasing the number of objects on the scene, error levels of 54% are achieved, the methods evaluated for separated give good results; objects detection with the similitude analysis and a good calibration, can achieved efficiency levels of 95%. High levels of error in the system evaluation are due to experimental place, because attempt to cover with a single camera eight accesses on the scene and this causes a large number of objects occluded. is proposed as future work to implement a video sensor at each access and use the ANOVA technique to analyze the similarity between the objects detected in a camera and other, or change the experiment to a single lane road.

(**Key words**: Accumulative Computation, Parzen Windows, ANOVA)

A mis padres

“Por su apoyo y atención en todo momento”

AGRADECIMIENTOS

A Dios por ser mi fuerza y el pilar de mi lucha constante.

A la Dra. Sandra Luz Canchola Magdaleno por haber sido mi directora de tesis, por compartir sus conocimientos y haber puesto la confianza en mí.

A la M.C. Verónica Josefina Soria Anguiano por haberme brindado su apoyo y sus conocimientos en la elaboración de este proyecto.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CUADROS

1. INTRODUCCIÓN.

El principal objetivo de la *Visión por computadora*, también conocida como *Visión artificial*, es simular la visión humana, no sólo el reconocer los contornos o niveles de color de las formas expuestas en una escena, sino de la percepción e interpretación automática del campo visual.

Hoy en día, la *Visión por computadora* es aplicada en una gran variedad de ramas de la ciencia, entre ellas se pueden mencionar algunas áreas: Cartografía y Sistemas de Información Geográfica (SIG), Metalurgia, Crecimiento de células, Reconocimiento de patrones psicológicos, Seguridad, Producción, Transporte, Reconocimiento de huellas, Percepción remota, Astronomía, Arqueología, Microscopía y muchas otras más.

La *Visión por computadora* no sólo es potencial por la gran capacidad para automatizar procesos que requieren actividades arduas y que consumen mucho tiempo, sino que también eliminan el error humano producido por factores como el cansancio o la distracción. Sin embargo cabe mencionar que la *Visión artificial* no siempre es la mejor solución, muchas veces resolver un problema a través de la visión requiere un nivel de inteligencia que una maquina no podría ofrecer. Por lo tanto es importante diseñar sistemas de *Visión por computadora* que exploten al máximo su capacidad.

El concepto de caracterización vehicular se define por la integración del conteo, clasificación y dirección de vehículos, lo cual es utilizado en el proceso del diseño carretero, ampliación y modernización de infraestructura, dichos procesos generan alto impacto económico y debido a esto sus inversiones requieren de estudios confiables y sólidos.

Según Egües (1964) “los estudios de volúmenes de tránsito se realizan siempre que se desea conocer el número de vehículos que pasan por un punto dado. Estos estudios varían desde los muy amplios en un sistema de caminos hasta recuentos en lugares específicos tales como puentes, túneles o intersecciones con semáforos. Las razones para efectuar estos recuentos son tan

variadas como los lugares en donde se realizan. Por ejemplo, los aforos se realizan para determinar la composición y volumen del tránsito en un sistema de carreteras; para determinar el número de vehículos que viajan en cierta zona o a través de ella; para evaluar el índice de accidentes; para servir como base en la clasificación de caminos; como datos útiles para la planeación de rutas y determinación de proyectos geométricos; para proyectar sistemas de control del tránsito; para elaborar programas de conservación; para establecer prioridades de construcción; para determinar el tránsito futuro y muchas otras aplicaciones".

Es importante mencionar que el concepto formal de volumen de tránsito también abarca a los peatones, sin embargo para el objetivo de estudio, se entenderá que el volumen se compone sólo de vehículos.

Ya que el tránsito vehicular es un elemento esencial para la toma de decisiones, control y construcción de obras de infraestructura, se han dirigido muchos esfuerzos para automatizar y mejorar el proceso de aforo y caracterización de vehículos. Se han probado distintas metodologías a lo largo del tiempo, se estudiaron los lazos inductivos (1994-2000), sensores de microondas (2002), magnetómetros (2004), sensores piezoeléctricos (2006) y visión por computadora (1996-2006). Sin embargo en la actualidad la robustez de la tecnología con base a la visión por computadora es la herramienta más usada, ya que a través de ésta, no sólo es posible hacer un conteo y clasificación, sino que también permite la extracción de parámetros como dimensiones, colas de espera, velocidad, distribución direccional, entre otros.

En año 2005 se publicó en Perú un documento denominado *"Arquitectura para el procesamiento de imágenes de tránsito vehicular"*, dicha publicación consistía en el desarrollo de una arquitectura que procesaba imágenes para obtener el volumen de tránsito. Se utilizaron las técnicas de filtros promedio, umbralización, dilatación morfológica y resta de imágenes. En el 2007 con la publicación *"Hacia un sistema automático de caracterización vehicular"* se logró no sólo automatizar el conteo de vehículos a través de las técnicas tradicionales, sino que también se desarrolló *Redes Neuronales Artificiales (RNAs),* que automatizan

la caracterización vehicular; los resultados fueron buenos y aunque no exactos se obtuvo un nivel de alfa de 0.05, lo cual es estadísticamente significativo. La falta de exactitud se debe a problemas de traslape, que son generados por las sombras de los propios vehículos, esto quiere decir que cuando varios vehículos van muy juntos el sistema lo cuenta como uno sólo, otro factor del problema, es en el cambio de carril, el sistema no logra contarlo y clasificarlo, de igual forma pasa cuando sale del borde de la visión de la cámara. Por ultimo en el 2008 con la publicación *"Extracción automática de características de vehículos en movimiento a partir de videos basada en la red neuronal de Kohonen"* se diseñó un sistema que extrae características de vehículos en movimiento empleando el procesamiento de imágenes y RNAs. El método de extracción de imágenes y procesamiento es el mismo que en las dos publicaciones anteriores, la diferencia se encuentra en la utilización de una *RNA de Kohonen*, modelo que soluciona los problemas de detección de movimiento en una secuencia de imágenes utilizando inteligencia artificial y manteniendo actualizada una imagen pivote.

El objetivo general de esta tesis consiste en desarrollar una metodología automática de conteo y clasificación de vehículos en un tramo carretero, por lo tanto se parte de una hipótesis que supone que, con la aplicación de un algoritmo en tiempo real es posible automatizar el conteo y clasificación de vehículos con un grado de certeza del 90% del aforo vehicular real, esto teniendo en cuenta que el grado de error radica en los fallos positivos (reconocer al objeto como vehículo cuando no lo es) y en los fallos negativos (no detectar al vehículo).

2. REVISIÓN DE LA LITERATURA.

Este apartado tiene como principal objetivo describir cada uno de los conceptos y métodos fundamentales para el desarrollo de la tesis, además se incluye un segmento donde se exponen proyectos relevantes y relacionados con el procesamiento digital de imágenes para aforar el tránsito vehicular.

2.1. Conceptos básicos.

2.1.1. Visión por computadora.

La *Visión por computadora*, también conocida como *Visión artificial,* es el concepto que surge en el intento de emular la percepción visual humana, ésta consiste en captar imágenes mediante dispositivos de captura para después aplicar técnicas de procesamiento de imagen y obtener las propiedades de algún objeto en específico.

La *Visión por computadora* tiene como principal objetivo generar información del mundo 3D que pueda ser interpretada por la computadora, así es posible procesarla y manipularla para obtener resultados específicos.

De acuerdo con el concepto y el objetivo principal de la *Visión artificial* expuestos anteriormente, se puede resumir que la *Visión por computadora* se concentra en mejorar la calidad de las imágenes y procesar la información de la escena para que una máquina la perciba de forma automática.

Hoy en día la *Visión artificial* es muy popular en un sin fin de áreas de aplicación, y se cree que esto es debido no sólo al éxito en la automatización de tareas, sino que también logra romper las barreras en muchas tareas en las que la capacidad física del hombre es insuficiente, se refiere a las actividades donde no influye el error humano, entre ellas se pueden mencionar el acceso a lugares peligrosos, monitoreos continuos y sin descanso, visión sin influencias de distracción o de salud, entre otros.

A continuación se mencionan algunas de las áreas donde se aplica la *Visión por computadora*:

- Procesos Industriales.
    - Control de calidad.
    - Control de inspección en los procesos de fabricación.
    - Seguridad.

  Ejemplo: Zambrano et al. (2007), presenta el desarrollo de una estación de control de calidad con base a la inspección y verificación del proceso de fabricación de piezas mecanizadas, el sistema analiza las piezas por medio de la Visión Artificial y con base a una plantilla o pieza patrón.

- Percepción remota.
    - Soporte en cartografía.
    - Monitoreo de recursos naturales.
    - Control de desastres naturales e incendios.
    - Clasificación de cultivos.
    - Aforos.
    - Detección de patrones psicológicos.
    - Detección de movimientos.

  Ejemplo: Nope et al. (2008), presenta un sistema que captura los gestos a través de una cámara Web, con base a éstos se aplican técnicas de reconocimiento de patrones que pueden ser comparadas con el objetivo de determinar qué características usadas son las más discriminantes.

- Robótica.
    - Control de inspección donde interactúan los robots de fabricación con un sistema de *Visión por computadora*.
    - Seguimiento a objetos.
    - Seguridad de vehículos.
    - Reconocimiento de objetos.

  Ejemplo: Prieto et al. (2010), presenta un sistema compuesto por un brazo robótico y la aplicación de técnicas de *Visión por Computadora*, el cual tiene

como objetivo facilitar las maniobras en una laparoscopía (técnica que consiste en hacer uso de un instrumento óptico para visualizar y analizar en tiempo real el contenido de la cavidad abdominal durante las cirugías), en dichos procedimientos, se identifica la imagen de la herramienta quirúrgica para obtener las coordenadas espaciales y la profundidad que harán que el robot siga automáticamente los movimiento del cirujano, y de este modo, reducir los riegos de error y disminuir los tamaños de incisión.

- Medicina.
    - Detección de tumores.
    - Detección y conteo de células cancerígenas.
    - Detección de deformidades.
    - Dispositivos para invidentes.
    - Monitoreo de pacientes.

  Ejemplo: Rueda et al. (2008), presenta el desarrollo de una herramienta que disminuye el tiempo de diagnóstico de tuberculosis pulmonar, esto a través de técnicas de segmentación por umbralización que permiten el conteo de bacilos (se usa para describir cualquier bacteria) y/o agrupaciones de esputo (secreción que se produce en los pulmones y en los bronquios) con tinción de Ziehl-Neelsen (técnica para la identificación de microorganismos patógenos).

- Muchas otras más.

2.1.2. Sistema de visión por computadora.

Un *Sistema de visión por computadora* es el arreglo de elementos electrónicos y eléctricos que en conjunto buscan capturar aquellos parámetros y características de interés traducidos a datos que pueden ser interpretados por una computadora, por lo tanto, el sistema hace posible la disponibilidad de información que puede ser procesada para resolver problemas y producir resultados específicos. Sus principales componentes son:

- Dispositivo de captura de imágenes:
    - Cámaras digitales (fotográficas y de video).

    - Microscopio.
    - Telescopio.
    - Escáneres.
    - Entre otros.
- Convertidor de señales (Digitalizador). Son los conversores analógico/digital (A/D), que se encargan en convertir las señales eléctricas a códigos binarios que pueden ser interpretados por la computadora para conformar las imágenes digitales.
- Computadora. La computadora es el elemento que almacenara las imágenes digitales entregadas por el dispositivo A/D, además de procesarlas para producir los resultados deseados.
- Manipulador. Es el dispositivo robótico que en ocasiones se presentará como necesario, la mayoría de las veces para seguir algún objeto de interés y ampliar la visión del sistema.

2.1.3. Procesamiento de imágenes.

El *Procesamiento de imágenes* es el conjunto de técnicas y métodos que tratan imágenes con el fin de extraer, mejorar y manipular sus parámetros y características. A continuación se describen dichas técnicas:

- Técnicas para mejorar una imagen. Son todas aquellas que buscan modificar la imagen ya sean para mejorar su calidad, procesarlas más rápidamente o prepararlas para eliminar parámetros que no son de interés. Algunos ejemplos son:
    - Los filtros espaciales.
    - La Binarización.
    - La Ecualización.
    - La dilatación morfológica.
    - La compresión.
    - Umbralización.
    - Entre otros.

- Extracción de características. Son todas aquellas que eliminan parámetros que no son de interés y que detectan aquellos que sí lo son. Algunos ejemplos:
    - Segmentación.
    - Detección de contornos y esquinas.
    - Entre otros.
- Manipulación de parámetros. Son todas aquellas que procesan los parámetros pertenecientes a los objetos extraídos y de interés para producir los resultados. Algunos ejemplos son:
    - Relación de distancias.
    - Relación de conectividad.
    - Frecuencias.
    - Transformaciones.
    - Interpolación.
    - Traslación.
    - Entre otros.

Es importante mencionar que dichas técnicas pueden cambiar de rol en determinado caso, por ejemplo una relación de distancias o de conectividad también puede utilizarse como técnica para mejorar una imagen.

2.1.4. Volumen de tránsito vehicular.

Como lo expresa Egües (1964), “Se denomina volumen de tránsito al número de unidades de tránsito que pasan por un punto dado en un período específico de tiempo”.

El volumen de tránsito es expresado por el número de vehículos en relación a una unidad de tiempo determinada, por lo regular en un día o en una hora. Las unidades de tránsito están compuestas por vehículos de todo tipo, así como de peatones, sin embargo para el objetivo del presente estudio, se entenderá que el volumen se compone sólo de vehículos.

2.1.5. Caracterización vehicular.

La clasificación vehicular es el proceso que identifica la tipología del vehículo a partir de sus características físicas, el resultado es un ordenamiento que es representado por una serie de categorías determinadas.

El concepto de caracterización vehicular se define por la integración de un conteo, clasificación y dirección de vehículos, lo cual es utilizado en el proceso del diseño carretero, ampliación y modernización de infraestructura. Se sabe que dichos procesos generan alto impacto económico y que debido a esto sus inversiones requieren de estudios confiables y sólidos.

2.1.6. Aforo vehicular.

El aforo vehicular es el proceso que se necesita realizar para determinar la composición y volumen de tránsito vehicular en un sistema de carreteras, existe una diversidad de métodos para desarrollar esta actividad, con base en su manera de realizarse se clasifican en manuales y automáticos.

Dependiendo del estudio que se desea realizar, es el tiempo de aforo requerido, en algunos casos para periodos cortos, por ejemplo una hora, para otros pueden ser 24 horas, una semana o hasta un mes. Existen carreteras en las cuales se llevan a acabo aforos permanentes, los 365 días del año, por ejemplo, en las casetas de cobro de autopistas y puentes de cuota, ver la ubicación de las estaciones de conteo de vehículos en la Dirección General de Servicios Técnicos (2014).

2.1.7. Métodos para aforar vehículos.

Existe una serie de métodos que se utilizan para aforar vehículos, la clasificación más sencilla podría ser englobada en dos términos, los métodos manuales y los automáticos, sin embargo, cabe mencionar que la factibilidad de cada uno de ellos depende de los datos que se desean obtener, el costo y los periodos de tiempo que se requieren abarcar para determinado estudio.

2.1.7.1. Métodos manuales.

Un método manual consiste en hacer uso de personal de campo conocido como aforadores de tránsito, este tipo de aforo se realiza cuando la información requerida no se puede obtener de los métodos automáticos, por ejemplo:

- Para probar la eficiencia de los dispositivos de aforo automático.
- Cuando las condiciones del tiempo interfieren con el correcto funcionamiento de los dispositivos automáticos.
- Los datos son poco comunes y la capacidad de los aforadores automáticos no es suficiente.

Los aforos manuales permiten obtener una clasificación por tamaño, tipo, número de ocupantes, direccionamiento, etc., las técnicas empleadas para desarrollar un estudio de aforo manual consisten en el uso de formatos especiales y estructurados, y del desarrollo de actividades variadas y relacionadas con el posicionamiento estratégico del grupo de aforadores.

Existe un método denominado *Automóvil en movimiento*, y aunque podría considerarse como una técnica manual su procedimiento es muy distinto a los tradicionales, permite determinar el volumen y la velocidad del tránsito viajando por un tramo carretero en un vehículo, con un operador y un observador. El observador registra los vehículos que transitan en sentido opuesto, los que lo rebasan y a los que rebasa, el vehículo recorre de un lado a otro el trayecto de estudio, este ciclo debe hacerse un determinado número de veces a una velocidad promedio respecto a los vehículos que transitan en el camino.

Para obtener el volumen de tránsito en una hora a través del método *Automóvil en movimiento* se utiliza el siguiente cálculo matemático: $V_H = \frac{60[M_e + (R-A)_{m8}]}{T_c + T_{m8}}$, donde V_H es el volumen de tránsito en una hora y en una dirección, M_e el número de vehículos encontrados en sentido opuesto, $(R-A)_{m8}$ es el número de vehículos que lo rebasan menos el número de vehículos

alcanzados en el mismo sentido, T_c es el tiempo o duración en minutos del viaje circulando en sentido contrario al flujo de estudio, T_{m8} es el tiempo o duración del trayecto circulando en el mismo sentido del flujo de estudios, y 60 es la constante.

A pesar de que existe una gran variedad de técnicas para aforar vehículos de forma manual, el interés de estudio de la tesis no requiere abarcar esta información, sin embargo es importante mencionar que estos métodos están generalmente limitados para periodos cortos o para lugares donde es la única forma posible de llevarlo a cabo.

2.1.7.2. Métodos automáticos.

Un dispositivo automático para aforar vehículos cumple generalmente con dos funciones, detectar o percibir el tránsito y realizar la recolección de la información del volumen de tránsito. Usualmente la detección se hace a través de impulsos eléctricos generados por un dispositivo y la recolección por medio de un registrador acumulativo. A continuación se describen algunos de los detectores para aforar vehículos:

- **Detector de neumáticos.** Es un tubo flexible instalado en el pavimento, un extremo está cerrado y el otro está conectado a un interruptor que se acciona bajo presión, al pasar las ruedas del vehículo sobre el tubo, estas desplazan un volumen de aire que crean una presión en el interruptor y accionan el registrador. El dispositivo tiene un bajo costo y es fácil de instalar, sin embargo es vulnerable a muchos riesgos, por ejemplo podría ser afectado por llantas con cadenas (usadas en clima con nieve), el frenado de un vehículo, vandalismo, robo, etc.
- **Contacto eléctrico.** Es una placa de acero cubierta por una capa de hule vulcanizado que contiene una tira de acero flexible, al pasar el vehículo sobre este dispositivo se cierra un circuito eléctrico, a través de éste se pueden hacer recuentos por carril.
- **Fotoeléctrico.** Este tipo de aforo consiste en hacer el registro de objetos por medio de un equipo fotoeléctrico, la presencia de un vehículo es detectada

cuando pasa a través de una fuente de luz y una fotocelda (dispositivo que distinga la ausencia de luz). El método no es conveniente para recuento de dos o más carriles y cuando se sabe que los volúmenes rebasan los mil vehículos por hora, esto es debido a la gran variación de las características de los vehículos y de los terrenos, es difícil determinar la altura de la fuente de luminosidad.

- **Radar.** Aforo realizado a través de un fenómeno conocido como “Efecto Doppler”, este consiste en la detección de objetos en movimiento que ocurre cuando la señal del radio cambia de frecuencia, el equipo electrónico utiliza el radar para comparar continuamente su frecuencia, siempre que exista una diferencia de frecuencias será detectado un vehículo. El método tiene la ventaja de que no sufre deterioro por la acción de los vehículos, sin embargo su costo inicial es mucho más alto que el de otros dispositivos.
- **Magnético.** Es a través de un campo magnético y las señales o impulsos originados por el vehículo, hay dos tipos de detectores magnéticos, los de tipo autogenerador y los que requieren de una excitación. Los primeros constan de un embobinado colocado en un tubo de fibra debajo de la superficie del pavimento, el impulso es ocasionado por la distorsión que amplifica el voltaje, por otro lado los detectores que necesitan de un impulso tienen dos embobinados con ajustes diferentes para anular uno de ellos bajo condiciones normales. Estos dispositivos no son vulnerables a las acciones de los vehículos, sin embargo las instalaciones subterráneas dificultan o anulan la posibilidad de hacer uso de ellos.
- **Ultrasónico.** Detecta los vehículos a través de una onda ultrasónica enfocada hacia el piso y recogida por una celda, al ser interrumpida se produce un cierre de un relevador (dispositivo que controla el estado de un interruptor mediante entradas eléctricas), este es el punto donde percibe al objeto. Este método detecta a los objetos en movimiento y a los estacionados, no es vulnerable a las acciones de los vehículos y es muy preciso, sin embargo es de alto costo inicial.

- **Infrarrojo.** Usa una celda de captación similar a la fotoeléctrica, los dispositivos infrarrojos pueden ser activos o pasivos, los activos tienen una fuente de energía infrarroja que es enfocada con un flujo hacia el piso y lo recoge por reflexión (fenómeno que incide cuando un rayo de luz es reflejado). Por otro lado, los infrarrojos pasivos detectan el calor producido por el objeto. En general los dispositivos infrarrojos no son afectados por las acciones de los vehículos, sin embargo sus costos iniciales son altos.
- **Visión por computadora.** Método que intenta emular la percepción de la visión humana, consiste en una o varias cámaras de video montadas en puntos estratégicos para detectar los vehículos que transitan sobre un tramo carretero, las técnicas asociadas a estos métodos no sólo permiten la detección y el conteo de vehículos, sino que también permiten obtener las características físicas de cada uno de ellos. Un *Sistema de Visión por Computadora* se puede resumir en el mejoramiento de las imágenes, la extracción de los objetos de interés, la modificación de los objetos extraídos y la obtención de los parámetros requeridos. Un ejemplo, Fayton y Moggia (2008).

Ya que se ha mencionado una serie de dispositivos que detectan a los vehículos, es importante también mencionar algunos registradores, los cuales son elemento esencial para obtener un método automático de aforo vehicular. A continuación se describen:

- **Indicadores visuales.** Son contadores acumulativos que registran y verifican las lecturas al principio y al final del periodo de recuento.
- **Cinta impresa.** Recoge los impulsos de los detectores y los almacena en un registro acumulativo para después imprimir los resultados en una cinta, los registro típicos trabajan en intervalos de 15 minutos o una hora, después de imprimir en cualquiera de los dos casos, el contador regresa a cero automáticamente.
- **Carta graficadora.** Usando un registrador circular (sobrescribe y reutiliza el primer archivo de registro cuando los datos ya se han escrito en la base de datos) se pueden registrar volúmenes de tránsito, de cero a mil vehículos para

intervalos de cinco a sesenta minutos en periodos de veinticuatro horas o de siete días.

- **Cinta para computadoras.** Es el uso de circuitos integrados con los que se pueden leer los volúmenes de tránsito vehicular con un contador digital.
- **Fotografía.** Se obtiene una secuencia de fotografías del tránsito vehicular en una zona particular para obtener un inventario periódico o continuo.

2.2. Estado del arte.

El estado de arte detectado para el desarrollo de esta tesis consiste en tres artículos relevantes que exponen su metodología para la automatización en el aforo de tránsito vehicular. A continuación se hará la descripción del proceso expuesto en cada una de las publicaciones:

Publicación no. 1. Crisóstomo y Yoshimoto (Perú 2005):

Se desarrolló una arquitectura de procesamiento de imágenes para el tránsito vehicular, consiste en aplicar una secuencia de algoritmos para la separación de cada uno de los vehículos presentes en la imagen y después hacer el conteo de los mismos.

Se realizaron las pruebas con un FPGA FLEX 10KE100 (dispositivo semiconductor que contiene bloques de lógica, su funcionalidad se puede programar) empleando imágenes a colores de 240 x 180 píxeles.

Todos los algoritmos para el procesamiento de imágenes fueron implementados en lenguaje C, las imágenes se tomaron a una altura mínima de 20 metros con el objetivo de minimizar los traslapes de vehículos.

La captura fue mediante una cámara NTSC (sistema de codificación y transmisión de televisión en color analógico) que podría alcanzar hasta 640 x 480 píxeles, sin embargo se capturaron imágenes de 240 x 180 píxeles debido a que no se requiere detalles de los vehículos, además de obtener menor cantidad de datos a procesar.

Las imágenes adquiridas son de 24 bits de color, esto para distinguir adecuadamente a todos los vehículos, en imágenes a escala de grises los objetos se confunden con la pista.

El proyecto consistía en obtener una imagen de la pista vacía, la cual será restada a cualquier imagen donde si existían vehículos, posteriormente se aplicaron una serie de algoritmos que permitían mejorar la imagen. Las técnicas utilizadas son las siguientes:

- Filtros pasa-bajos media.
- Resta de imágenes.
- Umbralización.
- Filtros media-binario.
- Dilatación morfológica.

Los parámetros obtenidos para determinar el nivel de umbralización fueron a través del histograma de la imagen resultante de la resta de imágenes. El elemento estructural para cualquier filtro u operación morfológica utilizado es una matriz de 3 x 3 (λ=1) con conectividad de 8.

A continuación se muestra la arquitectura desarrollada:

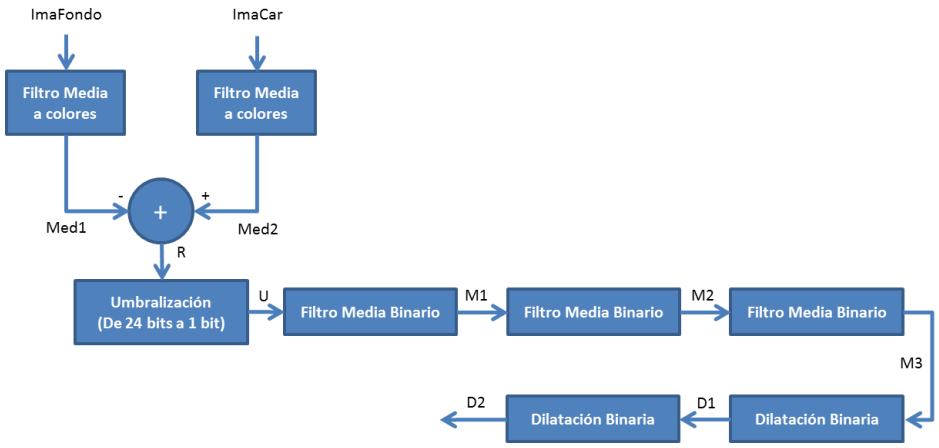


Figura 2-1 Arquitectura del proceso de Crisóstomo y Yoshimoto (2005)©. Imagen tomada del artículo bajo permiso de los autores.

Se analizó la imagen obtenida por la resta (con valores Red, Green, Blue) y se obtuvieron los valores de umbral adecuados (elemento estructural), con base a esto, los píxeles cuyos valores sean menores al elemento estructural serán el fondo (color blanco) y los restantes serán los objetos (color negro). Posteriormente se aplicaron tres etapas de la técnica *Filtro media-binario* para eliminar los puntos aislados, además se aplicaron dos etapas de la técnica *Dilatación morfológica* para eliminar agujeros de los objetos.

Al final se obtuvieron los vehículos aislados para después poder segmentarlos, etiquetarlos, calcular sus posiciones y dimensiones. La siguiente secuencia de imágenes extraídas del artículo, muestran algunos de los resultados obtenidos:

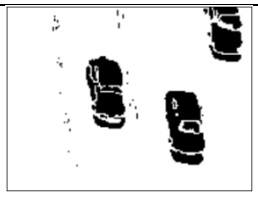
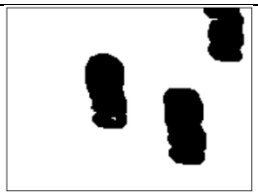
		
1. Pivote o fondo.	2. Con vehículos.	3. Resta de imágenes.
		
4. Umbralización.	5. Filtro Media Binario 1.	6. Filtro Media Binario 2.
		
7. Filtro Media Binario 3.	8. Dilatación 1.	9. Dilatación 2.

Tabla 2-1 Secuencia del proceso de Crisóstomo y Yoshimoto (2005)©. Imágenes tomadas del artículo bajo permiso de los autores.

Nota: Es importante mencionar que en este proyecto no se contemplan imágenes nocturnas o tomadas a menos altura de la establecida.

Publicación no. 2. Maldonado et al. (México 2007):

Se desarrolló un sistema automático de conteo y clasificación vehicular a través de técnicas de procesamiento de imágenes aplicadas a secuencias de video y la implantación de RNAs.

El proyecto se dividió en dos partes, la detección vehicular para obtener características que describan la presencia de un vehículo y la clasificación de vehículos a partir del análisis de las mismas. A continuación se muestra a través del siguiente esquema la metodología desarrollada:

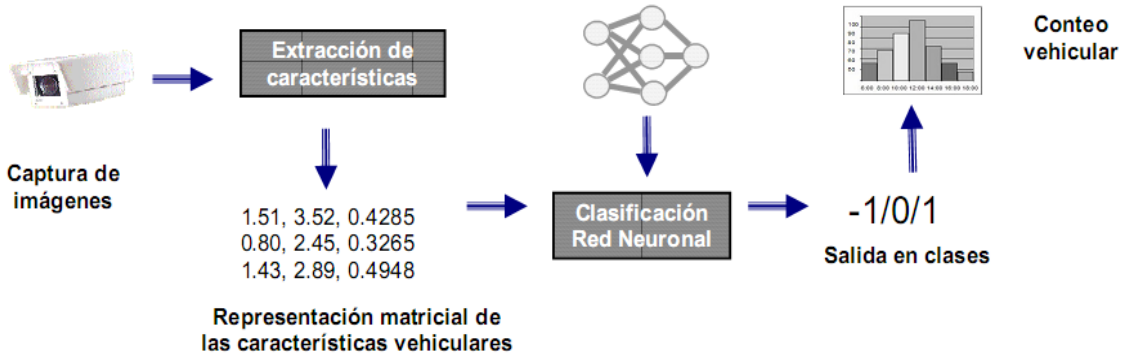


Figura 2-2 Arquitectura de Maldonado et al. (2007)©. Imagen tomada del artículo bajo permiso de los autores.

De acuerdo a la metodología expuesta en la **Figura 2-2**, se puede observar que las imágenes capturadas por una cámara de video se almacenan y procesan fuera de línea, sus parámetros son representados de forma conveniente por una RNA previamente entrenada para realizar la clasificación. El gran número de tamaños y formas de los vehículos dentro de una categoría es difícil de lograr usando parámetros simples, se eligió el uso de RNAs porque sabe que son especialmente robustas para la tarea de clasificación de información con ruido.

Para propósitos de este desarrollo, sólo se tomó un esquema de clasificación de vehículos pequeños, medianos y grandes. La clase vehicular pequeña está compuesta por automóviles identificados por sus fabricantes como ligeros, la mediana comprende todas las vans, SUVs, camionetas y pickups, etc., y la grande por autobuses, tráileres y semi-tráileres.

Se hicieron pruebas con una secuencia de video de cinco minutos con un flujo de tránsito moderado, el proceso de detección y conteo implico 1008 segundos, mientras la clasificación sólo tardo 28 segundos, haciendo un total de aproximadamente 17 minutos. Para comprobar la eficiencia de la arquitectura desarrollada, se comparó con un proceso de conteo y clasificación de forma visual (manual) sobre la misma secuencia de video, y aunque los resultados fueron aceptables, se detectaron problemas en el proceso automatizado a continuación se mencionan cuales:

- Duplicidad en detección de automóviles. (32 fueron detectado en dos ocasiones). Problema debido a la comparación de cuadros, dicha comparación es influenciada por las velocidades de los vehículos, a velocidades lentas es muy probable que sean detectados doblemente.
- Falsos negativos. (23 automóviles no fueron detectados). El problema se originó por varias razones: píxeles compartidos, por su cercanía al borde, por el número de píxeles y por parecer unidos a otro vehículo. De todos los problemas de este tipo, 14 se debieron a vehículos justo detrás de otro detectado anteriormente, el sistema considera que es el mismo vehículo y lo discrimina en la comparación de cuadros. El proceso que elimina a los objetos unidos al borde se realiza para evitar segmentar vehículos parcialmente ocultos, por lo tanto si un vehículo se detecta unido al borde de la carretera el sistema lo discrimina, sucedió en 7 ocasiones. El sistema elimina a todos los objetos menores a 300 píxeles, por lo tanto si un vehículo es detectado en un punto extremo del panorama del dispositivo de captura, es muy posible que sea menor a 300 píxeles y el sistema lo elimine, éste valor fue fijado empíricamente y ha mostrado resultado buenos, este problema sólo se

presentó una sola vez. Por último se identificó un error debido a la combinación de dos vehículos extremadamente juntos, si esto pasa el sistema lo detecta como uno sólo, problema presentado también una sola vez.

Los resultados erróneos presentados por la RNA son los siguientes: 9 de 211 vehículos pequeños fueron clasificados como medianos y 23 de 39 medianos como pequeños, dichos errores fueron debido a problemas de detección, por lo tanto 87.20% de los vehículos se clasificaron correctamente.

También se realizó una prueba con una secuencia de 90 minutos, sucedieron los mismos errores, además de la detección de motocicletas y bicicletas (35 fueron detectadas), ya que estos no fueron considerados en el estudio, 98.36% de los vehículos fueron detectados por el sistema, sin embargo 405 vehículos de 4,868 detectados se contaron doblemente y 521 no fueron detectados.

Partiendo de que no existe una regla definida para la obtención de los parámetros de una RNA y de que dependen en gran medida de la aplicación, se realizó un estudio a través de un diseño de experimentos para dos parámetros del clasificador vehicular, el tamaño del conjunto de entrenamiento y el número de neuronas en la capa oculta. El número de neuronas puede afectar el rendimiento del sistema si no se definen apropiadamente y el tamaño del conjunto de entrenamiento es sobre cuánto tiempo el usuario debe pasar entrenando al clasificador vehicular.

El tamaño del conjunto de entrenamiento se refiere al número de automóviles detectados en un lapso de tiempo de la secuencia de video, se consideraron los lapsos de 10, 20 y 30 minutos, a su vez los niveles considerados para el número de neuronas fueron de 8, 12 y 16. Por lo tanto se trabajó con un factorial de 3^2, quiere decir que dos variable a tres niveles y se realizaron cinco réplicas del mismo. Los resultados obtenidos son un rango de 75.9% a 76.4% de clasificación correcta, se determina que el tiempo es un factor que influye en el desempeño, caso contrario a lo que sucede con el número de neuronas en la capa

oculta, por lo tanto se identifica que al utilizar un tiempo de veinte a treinta minutos para el conjunto de entrenamientos se obtienen mejores resultados. La siguiente gráfica extraída del artículo representa los resultados obtenidos con el diseño factorial:

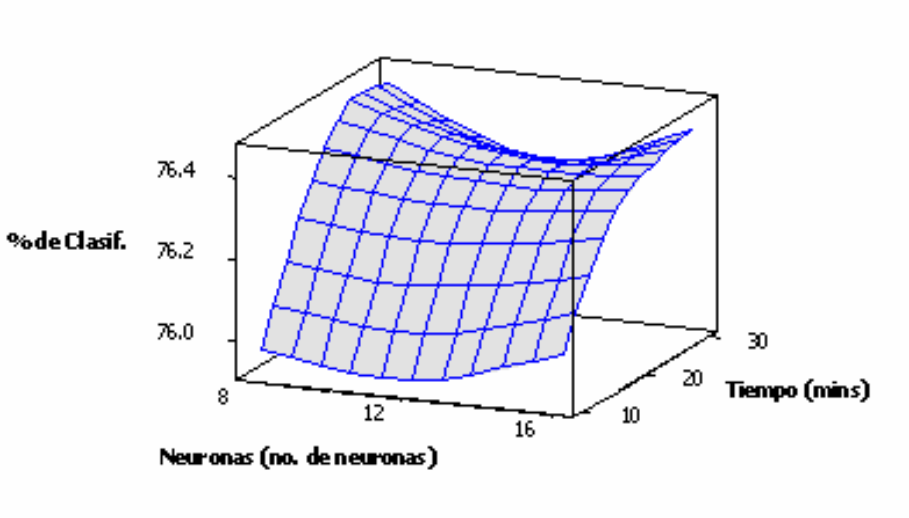


Figura 2-3 Resultados obtenidos por Maldonado et al. (2007)©. Imagen tomada del artículo bajo permiso de los autores.

Se dan algunas recomendaciones:

- Muchos de los problemas de la detección tienen su origen en el procesamiento intercalado de los cuadros que componen a la secuencia de video. El rastreo vehicular a todos los cuadros eliminará el problema, ya que se registrará la entrada y salida de los vehículos.
- La programación estructurada implica volver a un sistema computacionalmente costoso. La detección basada en programación orientada a objetos permitirá analizar cada uno de los cuadros de la secuencia de video al mismo tiempo que se despliegan, esto brinda la capacidad de ser utilizado en línea.

Publicación no. 3. Ferraz y Lameda (Venezuela 2008):

Se desarrolló un sistema para extraer las características de los vehículos en movimiento a través de procesamiento de imágenes y RNAs. La estructura del sistema se compone de dos subsistemas:

- **Detector de objetos en movimiento**. Se empleó una técnica de substracción de fondo en la que evoluciona a través de una máscara de actualización y computación acumulativa
- **Extractor de características**. Se implementó un filtro *Sobel* y aprendizaje a través de una RNA de *Kohonen*, la capacidad de dicha red neuronal permitió extraer los píxeles característicos de los objetos sin importar sus dimensiones, además de permitir establecer un número de píxeles fijo para entrada de un identificador de objetos en movimiento.

Detector de objetos en movimiento.

A través de la revisión teórica se determinó que el *Modelo de Interacción Lateral en Computación Acumulativa* era la mejor opción para desarrollar el subsistema de Detección de objetos en movimiento, ya que dicho modelo en teoría soluciona los principales problemas en la detección de objetos en movimiento, la computación acumulativa permite determinar la estabilidad de cada píxel, entendiendo cada píxel como fondo cuando permanezca estable cierta cantidad de veces. Posteriormente se requiere ejecutar filtros para disminuir el ruido causado por el dispositivo de captura. Por lo tanto, esta técnica supera los inconvenientes de la sustracción de fondo, sin embargo los falsos movimientos detectados por la variación de iluminación, movimiento de ramas y las sombras pueden causar problemas

A continuación se muestra la estructura del sistema detector de movimientos que logra solventar dichos problemas:

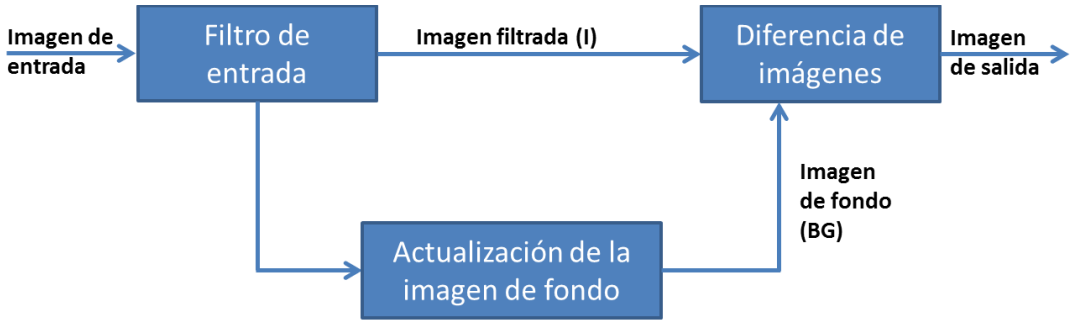


Figura 2-4 Sistema detector de movimiento de Ferraz y Lameda (2008)©. Imagen tomada del artículo bajo permiso de los autores.

Como podemos observar en la **Figura 2-4**, se realiza un filtro para acondicionar la señal de video, se utilizó un *Filtro Gaussiano* que suaviza la imagen convertida a escala de grises, a fin de disminuir el ruido ocasionado por el dispositivo de captura. La diferencia de imágenes realiza un filtrado de señal en la que deja pasar sólo los píxeles cuya diferencia con la imagen de fondo sea superior a un umbral (U1). Y por último, el bloque de actualización genera el fondo adecuado en cada instante de tiempo, permitiendo con esto que el bloque de diferencia de imagen sólo deje pasar objetos en movimiento.

El algoritmo de la **Figura 2-5** consiste en generar la imagen de fondo inicial como la primera imagen capturada, después el sistema aprende la imagen de fondo dependiendo la permanencia de estabilidad, esto quiere decir que cuenta el número de veces en que el píxel ha permanecido constante (es constante cuando la diferencia de dos imágenes consecutivas sea menor a un umbral *(U1)*), por lo tanto, cuando se supere el umbral de estabilidad *(U_{est})* el fondo se iguala a la imagen actual, esto se realiza con el objetivo de evitar que las regiones homogéneas sean consideradas como fondo. Para actualizar la imagen se utiliza una matriz de cambio de píxel *M(pixel)*.

A continuación se presenta el algoritmo utilizado para actualizar la imagen de fondo:

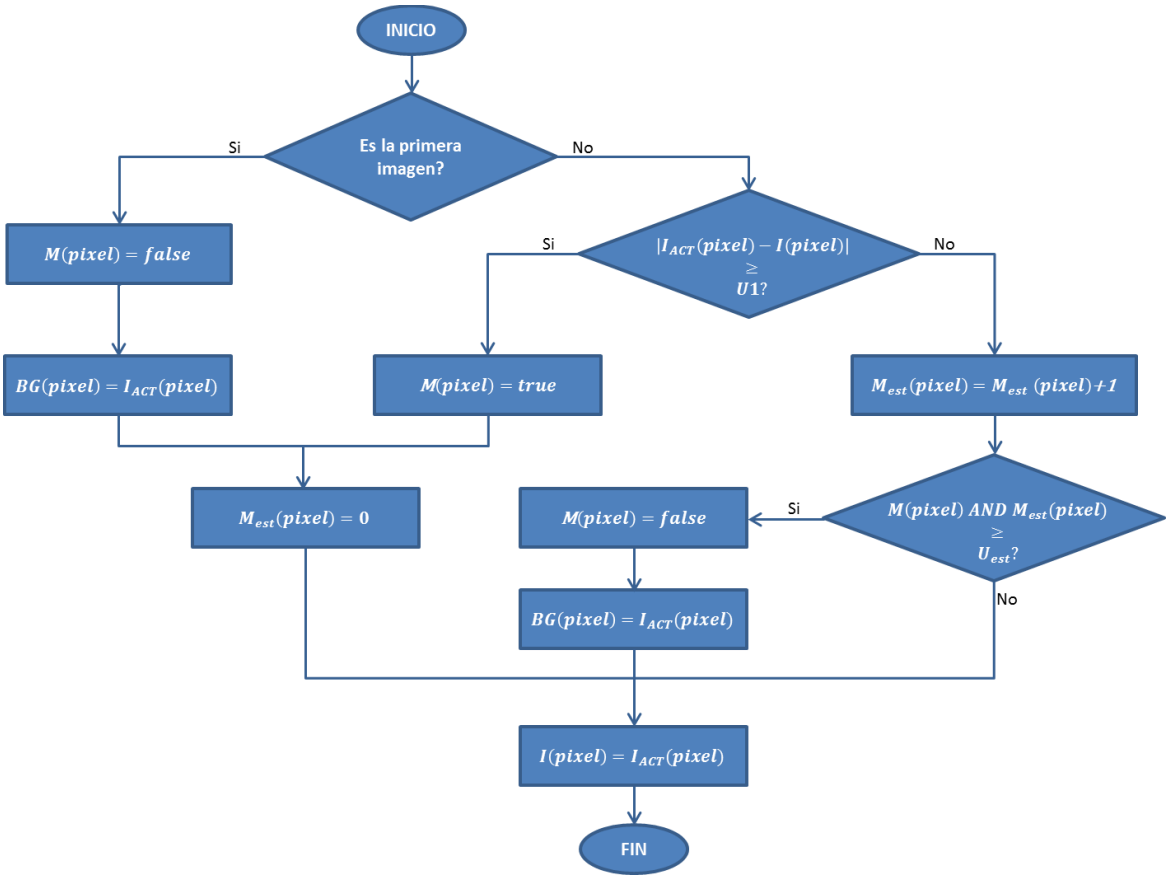


Figura 2-5 Algoritmo de actualización de Ferraz y Lameda (2008)©. Imagen tomada del artículo bajo permiso de los autores.

Extractor de características.

El extractor es el elemento que permitirá determinar sólo los rasgos más significativos de los objetos en movimiento, gracias a esto el clasificador será responsable de procesar una cantidad menor de datos.

Los contornos de una imagen son de gran utilidad para segmentar e identificar objetos, ya que éstos son zonas de píxeles en las que existe un cambio brusco de nivel de gris, las técnicas más usadas para la detección de contornos se basan en obtener el gradiente de las imágenes, ya que a través de éste se pueden obtener las variaciones del nivel de gris en píxeles consecutivos.

El ruido en las imágenes es el principal problema en las técnicas de extracción de contornos, por lo tanto se utilizan filtros de eliminación de ruidos antes de extraer los bordes. Teóricamente el **filtro Sobel** es empleado para determinar rasgos de los objetos, con esta técnica las fronteras de un objeto no se ven afectadas por las variaciones de intensidad de luz. Este filtro permite extraer los bordes de los objetos en movimiento, sin embargo la cantidad de píxeles extraídos es grande y variable entre un objeto y otro, por lo tanto, éste es uno de los retos más importantes del clasificador, ya que éste requiere una cantidad de píxeles que no sea variable entre los objetos para poder hacer la comparación en la base del conocimiento. Con esto se quiere decir que es necesario que se determine una cantidad fija de píxeles que represente los bordes de los objetos extraídos.

Se ha empleado una Red de *Kohonen* donde los píxeles del borde del objeto se usan como patrones de entrada, por lo tanto, cada neurona representa una característica principal del objeto. La RNA de *Kohonen* se caracteriza por poseer un aprendizaje no supervisado.

El modelo SOM (Mapas auto organizados) es el encargado de generar y entrenar la Red *Kohonen*, está compuesta por dos capas de neuronas:

- Capa de enterada. Formada por N neuronas, una por cada variable de entrada y se encarga de recibir y transmitir a la capa de salida los datos para procesar la salida.
- Capa de salida. Formada por M neuronas, ésta se encarga de procesar los datos y formar la capa de rasgos. Regularmente se organiza en forma de mapa dimensional, sin embargo se pueden organizar en una dimensión (cadena de neuronas lineal) o tres dimensiones.

Las conexiones entre las capas son siempre hacia delante, con esto se quiere decir que los datos se propagan desde la capa de entrada hacia la capa de salida. Por lo tanto, cada neurona de entrada *i* está conectada con cada neurona de salida *j* a través de un peso w_{ij}. Con base en esto, las neuronas de salida tienen

asociado un vector de peso w_j, el cual constituye el vector de referencia de la categoría representada por la neurona de salida *j*.

A continuación se presenta una arquitectura SOM en plano bidimensional:

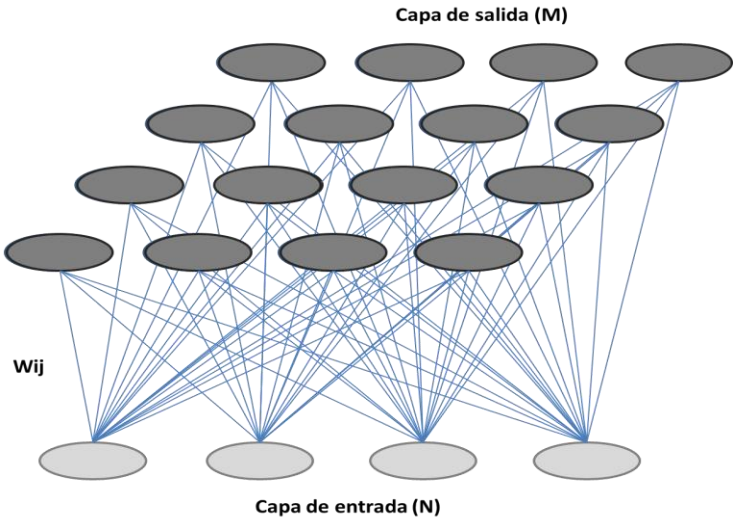


Figura 2-6 Mapa bidimensional de Ferraz y Lameda (2008)©. Imagen tomada del artículo bajo permiso de los autores.

En este modelo se dan las conexiones laterales de excitación e inhibición implícitas, esto quiere decir que entre cada una de las neuronas de la capa de salida, aun cuando no estén conectadas, existe cierta influencia entre sus vecinas. Esto es posible a través de una función de vecindad (marca la cercanía entre las neuronas). El funcionamiento formal de la RNA se describe a continuación:

1. Se presenta un patrón *p* de entrada *Xp* al SOM entrenado.
2. El patrón es transmitido directamente desde la capa de entrada hacia la capa de salida.
3. En la capa de salida, cada neurona calcula la similitud entre el vector de entrada *Xp* y su propio vector de peso W_j o vector de referencia, según cierta distancia o similitud establecida.
4. Por último, se declara vencedora la neurona que tenga el vector de peso más similar al vector de entrada.

El funcionamiento expresado anteriormente es representado por la siguiente expresión:

$$y_{pj} = \begin{cases} 1 & si \quad \min\left|\overline{X}p + W_j\right| \\ 0 & para \quad el \quad resto \end{cases}$$

Donde y_{pj} es el grado de activación de las neuronas de salida en función del resultado de la competición, significa que el valor 1 representa a la neurona vencedora y 0 a la no vencedora, $\left|\overline{X}p + W_j\right|$ representa la medida de similitud entre el vector de entrada Xp: $x_{p1},\ldots, x_{pi},\ldots, x_{pN}$ y el vector de referencia o de peso W_j: $w_{j1},\ldots,w_{ji},\ldots,w_{jN}$ de las conexiones entre cada una de las neuronas de entrada y las de salida.

El aprendizaje de una RNA de *Kohonen* trata de establecer las diferentes categorías que servirán para clasificar nuevos patrones de entrada a través de la gestión de un conjunto de patrones de entrenamiento. A continuación se describe el proceso de forma simplificada:

1. Se presenta y procesa el vector de entrada.
2. Se establece como neurona vencedora cuyo vector de peso o de referencia sea el más parecido al vector de entrada.
3. El vector de referencia se modifica de forma que se parezca un poco más al vector de entrada. Esto con el objetivo de que dicha neurona responda en el futuro aún con más intensidad.
4. El proceso se repite para el conjunto de patrones de entrada, los cuales son presentados repetidamente, esto quiere decir que los vectores de referencia sintonizarán con uno o varios vectores de entrada.
5. Si existen más patrones de entrenamiento que neuronas de salida, más de un patrón deberá asociarse con la misma neurona (misma clase), por lo tanto los pesos que componen al vector de referencia se obtienen como promedio (centroide) de dichos patrones de la misma clase.

A continuación se describen dos funciones para calcular la similitud de vectores, las cuales consiguen identificar a la neurona vencedora:

- Distancia euclidiana. Dos vectores serán más similares cuanto menor sea su distancia. Su representación matemática es:

$$\min\left\|X_p - W_j\right\| = \min\sum_{i=1}^{N}(x_{pi} - w_{ji})^2$$

- Correlación o producto escalar. Dos vectores serán más similares cuanto mayor sea su correlación. Su representación matemática es:

$$\min\left\|X_p - W_j\right\| = \max\sum_{i=1}^{N} x_{pi} w_{ji}$$

El modelo SOM crea relaciones entre las neuronas próximas de la red, esto a través de una función llamada *zona de vecindad* que define el entorno que rodea a la neurona ganadora actual, su objetivo principal es que no sólo se actualice la neurona ganadora, sino que también las neuronas vecinas, con base a esto, las neuronas vecinas se sintonizan con patrones similares.

Ya identificada la neurona vencedora a través de los criterios de similitud, se modifica su vector de peso asociado y las neuronas vecinas, esto mediante la siguiente regla de aprendizaje:

$$\Delta W_{ij}(n+1) = \propto(n)(x_{pi} - W_{j^*1}(n))$$

Para: $j \in Zona_{j^*}(n)$

donde:

n, Es el número de ciclos o iteraciones.

$\propto(n)$, Es la tasa de aprendizaje (se inicializa con un valor entre 0 y 1).

$Zona_{j^*}(n)$, Es la zona de vecindad que tiene la neurona vencedora.

j^*, Es la neurona vencedora.

En este modelo, el vector de referencia es ajustado cada vez que un patrón de entrenamiento se presenta, sin embargo es posible acumular los incrementos calculados para cada patrón de entrenamiento para después actualizar los pesos con base a un promedio de incrementos acumulados, con esto se evita la oscilación (moverse de un lado a otro) de patrones y la aceleración de cambios bruscos entre los pesos de la red.

A continuación se describen las dos fases que se implementaron en el proceso general de aprendizaje:

- Fase 1. Se organizaron los vectores de peso en el mapa de red, comenzando con una tasa de aprendizaje y tamaños de vecindad grandes, esto con el objetivo de ir reduciendo su valor a medida que avanza el aprendizaje.
- Fase 2. Se persigue un ajuste fino del mapa de red, de modo que los vectores de peso se ajusten más a los vectores de entrenamiento. El proceso es similar al anterior, sólo que es más largo. Se toma una tasa de aprendizaje constante y con un valor pequeño (por ejemplo 0.01) y un radio constante de vecindad igual a 1.

La siguiente figura muestra la estructura de la red Kohonen:

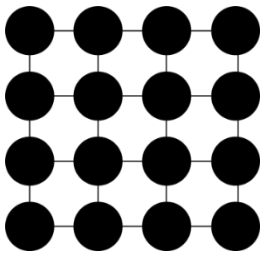


Figura 2-7 Estructura de la red de Ferraz y Lameda (2008)©. Imagen tomada del artículo bajo permiso de los autores.

La red se basa en una matriz de $4x4$, lo cual es determinado por la geometría de los objetos a clasificar, también es importante señalar que cada neurona consta de dos pesos que representan las coordenadas x e y en función a la su posición en el plano.

La siguiente figura representa el modelo de extracción de características:

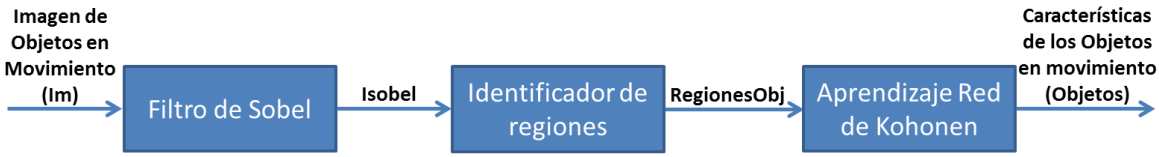


Figura 2-8 Diagrama del extractor de características de Ferraz y Lameda (2008)©. Imagen tomada del artículo bajo permiso de los autores.

Se puede observar que el modelo de la **Figura 2-8** aplica un filtro *Sobel* a toda la imagen, por lo tanto fue necesario agregar un bloque *Identificador de Regiones* para determinar la región que pertenece a cada uno de los objetos en movimiento, esto con el objetivo de entrenar a la red *Kohonen* con píxeles que pertenecen a un mismo cuerpo.

Actividades y resultados obtenidos en la evaluación de la propuesta:

- Se creó un prototipo bajo la plataforma Windows y lenguaje C.
- Se capturaron videos desde una WebCam a diferentes horas del día.
- Se agregaron patrones de entrenamiento y se fue ajustando los umbrales hasta obtener la respuesta más adecuada. El prototipo final permitía ajustar los umbrales correctos a través de un software que muestra la imagen de entrada, imagen de fondo generada, imagen al aplicar la substracción de fondo, las regiones de los objetos detectados y los píxeles característicos de cada objeto.
- Se extrajeron los objetos de tres videos bajo diferentes condiciones de iluminación, esto a través del software mencionado en el punto anterior. Esto se hizo con el objetivo de obtener las primeras características de clasificación de vehículos, las cuales se almacenaron en el archivo de patrones.
- Los umbrales para la substracción de fondo y para generar la imagen pivote, tienen el mismo valor, ya que éstos cumplen el mismo propósito y juntos permiten determinar cuándo un píxel ha cambiado. Para la selección de este valor se realiza la comparación del resultado de variar estos umbrales.

A continuación se muestra una imagen donde se han aplicado tres niveles de umbral para obtener la imagen de fondo:

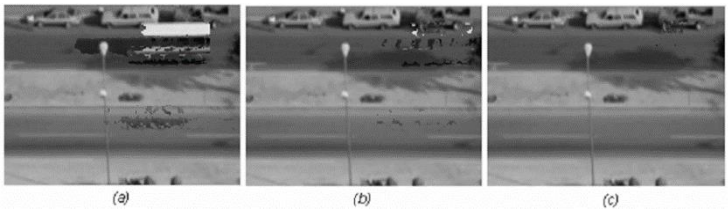


Figura 2-9 Extractor con valores (a) $U_{est}= 2$, (b) $U_{est}= 5$ y (c) $U_{est}= 10$ de Ferraz y Lameda (2008)©. Imagen tomada del artículo bajo permiso de los autores.

- Se aplicaron los filtros *Sobel* necesarios para extraer los bordes de los objetos, a continuación en la siguiente figura se presentan algunos resultados:

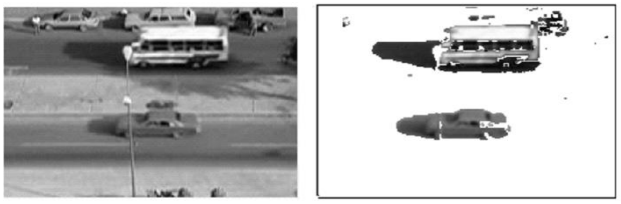


Figura 2-10 Aplicación del filtro Sobel de Ferraz y Lameda (2008)©. Imagen tomada del artículo bajo permiso de los autores.

- Para extraer los vehículos es importante el cálculo de vecindad de píxel, ya que de esta forma es posible determinar cuándo un píxel pertenece a un objeto. A continuación en la siguiente figura se muestran los resultados obtenidos al aplicar tres valores de vecindad:

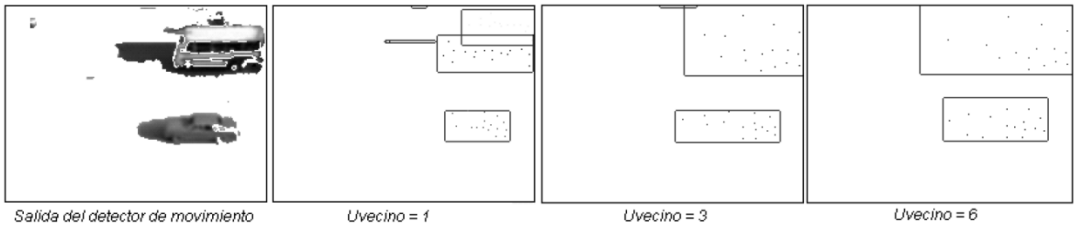


Figura 2-11 Aplicación de la vecindad para extraer objetos de Ferraz y Lameda (2008)©. Imagen tomada del artículo bajo permiso de los autores.

3. METODOLOGÍA.

El objetivo de este apartado es describir a detalle cada uno de los métodos y pasos que se siguieron para desarrollar el algoritmo propuesto en esta tesis, se presenta cada una de las técnicas utilizadas y se determina la eficiencia de cada una de ellas, en esta sección también se explican las arquitecturas, modelos y estructuras utilizadas para cumplir con el propósito planteado.

3.1. Descripción del lugar experimental.

La experimentación del algoritmo se conformó en un crucero ubicado en **Constituyentes/Pasteur, Querétaro, México**; la infraestructura del sitio permitió instalar el equipo de visión por computadora a una altura que ayuda en la eficiencia del proceso de detección de objetos en forma aislada, Crisóstomo y Yoshimoto (2005) aseguran que entre más alto este instalado el visor, es más fácil evitar los traslapes de vehículos y por consecuente las fusiones entre ellos. A continuación se presenta una fotografía de dicha intersección:

Figura 3-1 Intersección Constituyentes/Pasteur. *Extraída de Google Maps*.

3.2. Equipo experimental.

El equipo experimental es un sistema integrado que consta de una cámara de video y una PC de escritorio, a continuación se mencionan sus características principales:

- PC.
    - Procesador de tres núcleos a 2.93 GHz.
    - 4 GB de memoria RAM.
    - Tarjeta de video de 512 MB.
- Cámara de video.
    - Sensor CCD (Charge Coupled Devices).
    - 600 Mega píxeles de resolución espacial.
    - Sistema de estabilización y reducción de ruido digital.
    - Tarjeta capturadora.
    - Lente varifocal autoiris de 12 milímetros para ajustar automáticamente el ángulo de visión.

3.3. Lenguaje de programación.

Existe una serie de lenguajes de programación con los que se pueden desarrollar aplicaciones de Visión Artificial, sin embargo es importante detectar las ventajas y desventajas de cada uno de ellos.

Sin duda alguna el desarrollo de un software en **Lenguaje Ensamblador** garantizará un rendimiento óptimo, no obstante se requieren conocimientos muy avanzados y también mayor tiempo de desarrollo. Por otro lado, con lenguaje **C** se puede realizar el mismo algoritmo, obtener rendimiento y soluciones a corto plazo, esto se debe a que **C** provee de herramientas de abstracción y librerías como **OpenCV** que facilitan la implementación de la Visión artificial, características de las que el lenguaje ensamblador carece. Según Álvarez (2010), **C** también permite cierto desarrollo a bajo nivel, similar a **Ensamblador**; ventaja que puede generarse al combinar sus capacidades ofrecidas por ser un lenguaje de alto nivel,

además de que a diferencia de **Ensamblador**, se puede proporcionar el control de tipos de datos y la facilidad en la ampliación y modificación.

Existen otros lenguajes como **Matlab**, que es una solución de alto nivel y es muy usado para el procesamiento de frecuencias e imágenes, además la mayoría de sus funciones son implementadas en **C**, el único inconveniente es que no es una solución gratuita y que sacrifica un poco de rendimiento en comparación con el leguaje nativo **C**.

Octave, **Scilab**, **Sripy sobre Python**, etc. son soluciones de alto nivel que facilitan el trabajo, sin embargo y a pesar de que son de licencia gratuita el rendimiento es sacrificado notoriamente en comparación con **C**.

Ya detectado las ventajas y desventajas de los lenguajes de programación más usados para la visión artificial, se optó por usar el lenguaje **Java**, ya que según Vélez (2013), es de licencia gratuita, tiene bibliotecas potentes para la visión artificial (como **JAI**, **ImageJ** o **JavaCV**), es el lenguaje más utilizado por todo el mundo, es multiplataforma (se compila en código byte, lo que significa que es independiente y se ejecuta en una máquina virtual) y de fácil integración a páginas Web a través de *applets* o *servlets*. También es importante mencionar que a pesar de que **C** proporciona mayor rendimiento que cualquiera de los otros lenguajes mencionados anteriormente, **Java** podría ofrecer un rendimiento similar si se hiciera uso del equipo adecuado y se explotara al máximo su robustez al reservar y liberar la memoria (a través del elemento *Garbage collector*), y al diseñar aplicaciones *multi-hilo* (ejecución de tareas en paralelo), además de que este lenguaje elimina los conceptos complicados de **C/C++** al implementar herencia y apuntadores.

3.4. Librerías.

En el apartado anterior ya se han mencionado algunas de las librerías **Java** para la visión artificial, hablamos de **Java** porque se ha identificado que este lenguaje cuenta con las herramientas necesarias para cumplir con el objetivo de la tesis.

A continuación, se dará una breve descripción de aquellas librerías **Java** que podrían agilizar el desarrollo de la aplicación propuesta:

- **JMF (Java Media FrameWork)**. Es un API que proporciona las herramientas para capturar, almacenar y procesar datos multimedia. Sus principales características son:
    - Reproducir ficheros multimedia en *applets* y aplicaciones.
    - Reproducir flujos multimedia en tiempo real a través de la red.
    - Capturar y reproducir audio y video.
    - Aplicación de efectos como compresión, descompresión, conversión de formatos, etc.
- **ImageJ**. Es un software libre de procesamiento digital de imágenes desarrollado en Java, a continuación se mencionan sus principales características:
    - Código abierto.
    - Permite la extensibilidad a través de plugins java y macros grabables (macroinstrucciones).
    - Puede ser incorporado al editor y compilador Java.
    - Compatible con pilas y es multiprocesos, esto quiere decir que puede leer un archivo de imagen mientras se realizan otras operaciones en paralelo.
    - Procesa imágenes de 8 bits, 16 bits, 36 bits y RGB.
    - Ejecuta técnicas como suavizado, detección de bordes, filtrado, umbral, brillo, contraste, medición de área, medias, desviación estándar, longitudes, ángulos, generar histogramas, generar gráficos, etc.
- **JAI**. Es un API que proporciona un conjunto de interfaces que permiten una simple manipulación de imágenes en aplicaciones y en *applets*. Sus principales características son las siguientes:
    - Proporciona un alto rendimiento.
    - Es independiente de plataforma.
    - Creación de histogramas en tres colores.

- Trasforma imágenes de forma avanzada y sencilla.
- Contiene técnicas avanzadas como métodos de área, de color, geométricos, de extracción de bordes, etc.
- Con pocas líneas se pueden aplicar los métodos antes mencionados.
- No es necesario realizar operaciones matriciales.
- Es compatible con Java 2D.

- **JavaCV**. Es un contenedor que llama a las funciones nativas de **OpenCV**, el cual es un conjunto de bibliotecas que permite desarrollar aplicaciones de Visión Artificial, contiene un catálogo de funciones que incorporan cientos de algoritmos de procesamiento de imágenes y visión por computadora en tiempo real. Es una herramienta libre y de código abierto que puede ser explotado desde **JavaCV**. A continuación sus principales características:
  - Fácil de emplear métodos en paralelo en múltiples núcleos.
  - Fácil de emplear la calibración geométrica.
  - Tiene biblioteca multiplataforma.
  - Utiliza rutinas optimizadas de propiedades aceleradas.
  - Incluye árboles de decisión de aprendizaje.
  - Incluye Redes Neuronales Artificiales.
  - Realiza todo tipo de procesamiento de imágenes.
  - Algunas de sus áreas de aplicación son: procesamiento 2D y 3D, estimación de movimiento, reconocimiento facial, identificación y segmentación de objetos en movimiento, etc.
- **JCortex**. Es un *Framework* que permite desarrollar y utilizar redes neuronales artificiales en proyectos **Java**, esta herramienta se divide en **JCortex** para los desarrolladores y el entorno gráfico **JCortexBuilder** para **JCortex**. A continuación sus principales características:
  - Permite personalizar a distintos niveles de profundidad en las RNAs ya establecidas.
  - Permite añadir nuevos modelos.
  - Enmascara las bases matemáticas.
  - Es software gratuito y de código abierto.

- Gestiona entrenamientos simultáneos.
- Puede almacenar la red en formato XML.

3.5 Arquitectura preliminar.

Se parte de una arquitectura general que evoluciona a través de las pruebas y experimentación, esto con el objetivo de mejorar los modelos y técnicas propuestas por los investigadores citados en el apartado ***Estado del arte*** de este documento.

Es importante mencionar desde este punto y en relación con la revisión de la literatura expuesta anteriormente, que la aportación de esta tesis consiste en proponer un algoritmo fácil de entender y de manipular, además de hacer uso de secuencias de programación de bajo nivel que se puedan adecuar a cualquier tipo de lenguaje que soporte el procesamiento digital de imágenes y de video.

A continuación se presenta una arquitectura preliminar del sistema:

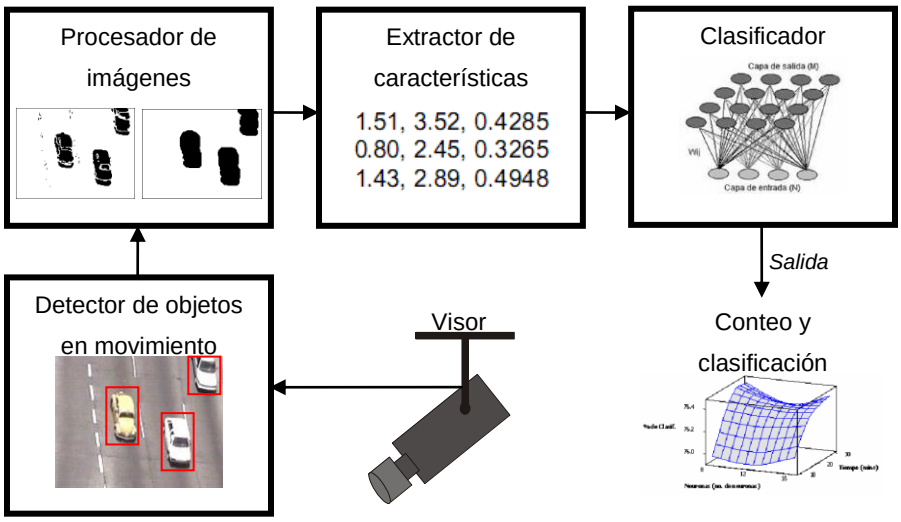


Figura 3-2 Arquitectura preliminar del sistema de visión por computadora propuesto.

La arquitectura de la **Figura 3-2** consiste en la detección de los vehículos en movimiento a través de una cámara de video, ya detectados los objetos que se

mueven se procede a la aplicación de técnicas de tratamiento de imágenes para eliminar ruidos y aislar los objetos de interés, después se extraen los datos característicos de cada uno de los cuerpos, esta información es presenta a una Red Neuronal Artificial, por ejemplo, la red Kohonen de Ferraz y Lameda (2008), donde los píxeles del borde son dichos rasgos de entrada y cada neurona es una característica principal de objeto; conforme se va transmitiendo el patrón de entrada, se forma un mapa de atributos asociado a un vector de pesos que al final permite determinar del tipo de vehículo. La salida de dicho sistema será el conteo y clasificación vehicular. En los siguientes capítulos se detallan cada una de los elementos del diseño propuesto.

3.6 Detector de objetos en movimiento.

El detector de objetos en movimiento consiste en el procesamiento de las secuencias de imágenes que busca determinar si un objeto se está moviendo, se entiende por secuencia de imágenes a la participación de dos o más imágenes que son parte de una escena determinada. Por lo tanto, cuando un objeto se mueve, se genera una variación espacio-temporal que por lo regular es poco significativa. Según Pajares y de la Cruz (2008), pueden ocurrir cambios bruscos, pero esto se debe a que el visor se encuentra muy alejado (ej. Imágenes tomadas por un satélite) o se requiere hacer un procesamiento de secuencias tomadas en distintos instantes de tiempo (meses o años), sin embargo para el tema de estudio de esta tesis el visor se encontrará a menos de 30 metros de altura del área de interés, además de que el proceso para extraer las características que se desean obtener por cada objeto requieren de periodos muy cortos.

3.6.1 Técnica de detección de movimiento.

Se establece una primera división de los métodos de detección de movimiento, movimiento 3D y movimiento 2D. Fernández et al. (2000) mencionan en uno de sus artículos que las técnicas 3D intentan resolver las ecuaciones de proyección usando directamente las características entre las tramas y las técnicas 2D estiman el flujo óptico.

El método de detección 3D es muy complejo, ya que requiere de técnicas de proyección 2D y deducción de movimiento, lo cual implicaría mucho tiempo de análisis y de recursos computacionales que limitarían el rendimiento de la aplicación que resolverá el problema planteado. Por lo tanto, esta tesis hará uso de técnicas 2D que permitan la estimación del movimiento, que haga posible la extracción de las características dimensionales de cada objeto detectado y permita el procesamiento en tiempo real.

Una clasificación básica de técnicas de detección de movimiento 2D es expuesta por Fernández et al. (2000), dicha categorización se basa en dos filosofías: la utilización directa de flujo óptico y el cálculo de diferencias entre las tramas de una secuencia de imágenes. A continuación se describe:

- **Basado en el gradiente.** En este método se usa directamente la ecuación del flujo óptico y determina los cambios de niveles de intensidad de las imágenes. A través del gradiente se mide la rapidez en que los píxeles cambian en distancia y en dirección.
- **Basado en correlación de regiones.** Este método a diferencia del basado en el gradiente, considera las regiones de la imagen en lugar de estimar el movimiento píxel a píxel, en resumen, divide las imágenes en una serie de regiones de igual tamaño, cada una de estas divisiones busca en su siguiente la posible correlación de vecindad.
- **Basado en el dominio de frecuencias.** Explota la invariancia (es la propiedad de una entidad que al aplicarle un conjunto de transformaciones, sigue siendo indistinguible a la original) en el desplazamiento y en las propiedades de simetría de la transformada de Fourier, existen dos clases de este tipo: la técnica Basada en Fases que determina el movimiento a través del dominio de frecuencias y la técnica Basada en Energía que lo determina con la distribución de energías en las frecuencias y sus orientaciones espacio-temporales.
- **Basado en el enfoque bayesiano.** A través de técnicas estocásticas (probabilidades dinámicas o cambiantes) y modelo explícito de propiedades estructurales del campo de movimiento, se usa un criterio máximo a posteriori

(MAP) para maximizar la probabilidad del movimiento a través de la observación de intensidad de la próxima *trama*. En resumen este método observa un resultado suponiendo que la realidad sea de una manera determinada (como una hipótesis).

Aunque existe una serie de métodos para detectar los objetos en movimiento con aplicación al tema de tesis, se optó por utilizar el *Modelo de Interacción Lateral en Computación Acumulativa*, ya que Ferraz y Lameda (2008) demostraron que esta técnica soluciona los principales problemas en la detección de movimiento en secuencias de imágenes a través de sus paradigmas de Inteligencia Artificial.

A continuación se presenta el modelo propuesto para la detección de movimiento:

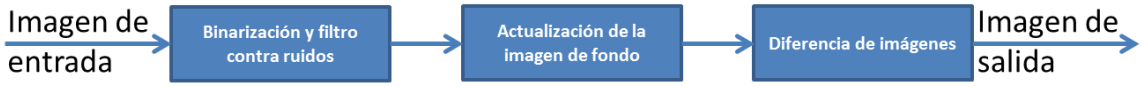


Figura 3-3 Diagrama de detección de movimiento.

El primer bloque del modelo de la **Figura 3-3** consiste en disminuir el ruido causado por el sensor del sistema y convertir la señal a escala de grises, el bloque medio se encarga de actualizar el fondo adecuado en cada instante de tiempo (ver en el apartado **3.6.2 Computación acumulativa**) para que el último bloque use la imagen pivote adecuada, calcule su diferencia y detecte sólo los objetos en movimiento. En resumen los bloques del modelo tienen la responsabilidad de:

- Acondicionar la señal de video capturada.
- Calcular el fondo adecuado.
- Separar los objetos de interés.

3.6.2 Computación acumulativa.

Ferraz y Lameda (2008) demuestran que a través de la computación acumulativa se puede generar un fondo adecuado para que el proceso de diferencia de imágenes sólo capte los objetos en movimiento, esta técnica supera los inconvenientes de falsos movimiento detectados por variación de iluminación, movimientos de objetos que no son de interés y las sombras.

A continuación se presenta el algoritmo propuesto para actualizar la imagen de fondo:

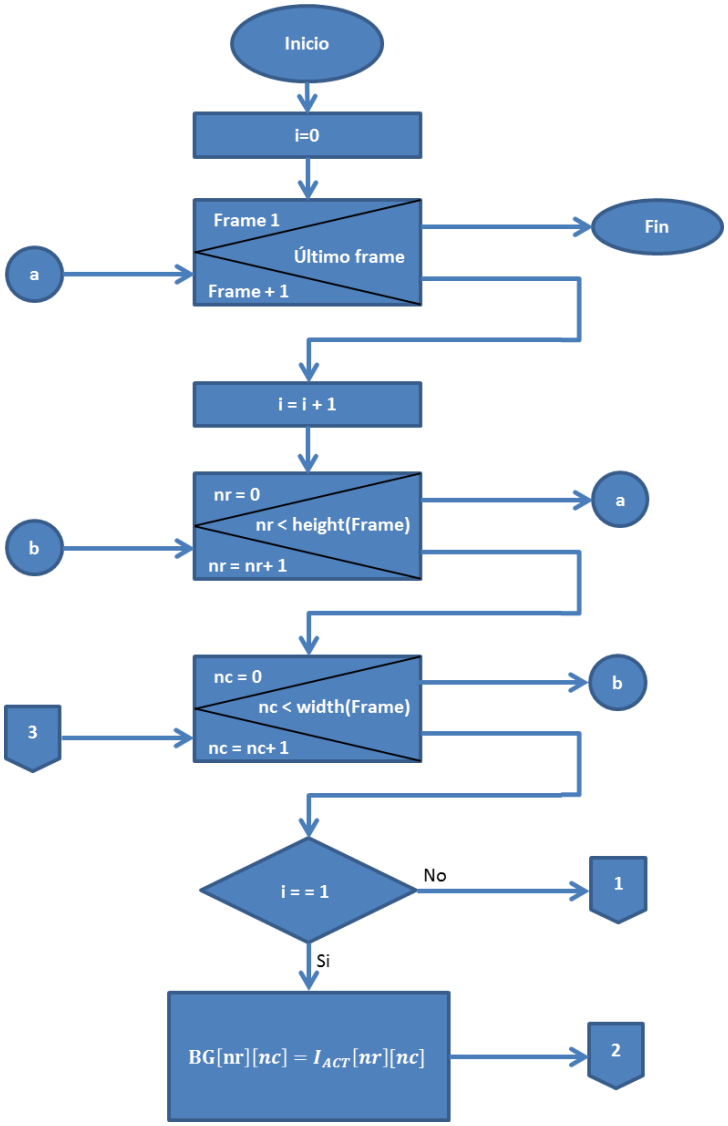


Figura 3-4 Flujo del algoritmo que actualiza el fondo, parte 1/2.

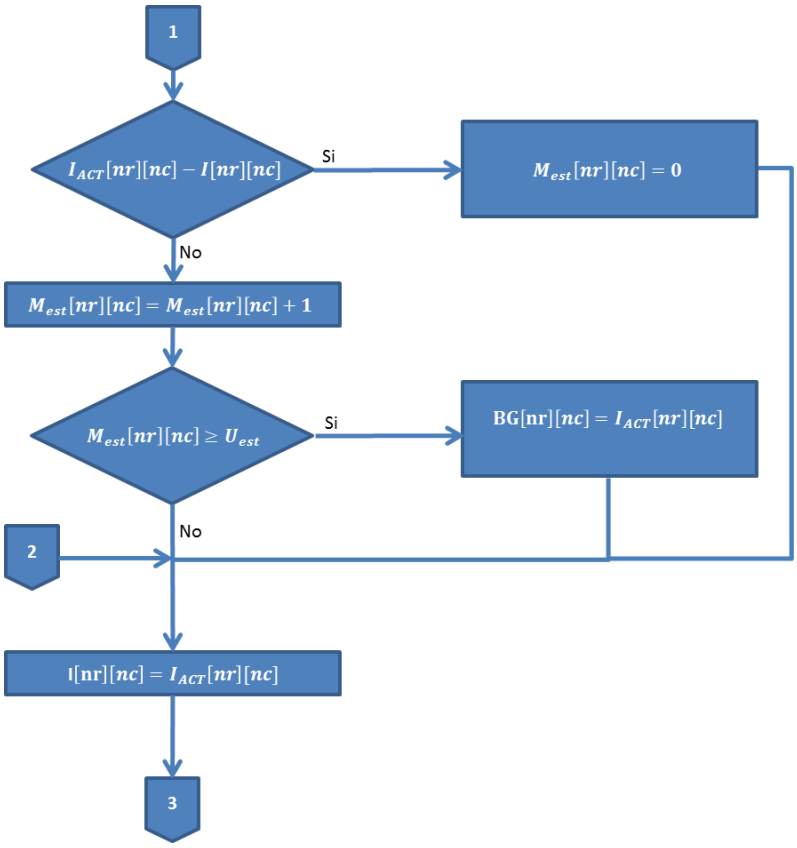


Figura 3-5 Flujo del algoritmo que actualiza el fondo parte 2/2.

La computación acumulativa es un modelo secuencial que se representa por su valor de permanencia o estabilidad (U_e), consiste en determinar cuándo un objeto se mantiene estático, esto es posible a través del procesamiento de cada píxel de los *frames* de la secuencia de video, la diferencia de cada píxel de un *frame* actual (I_{act}) con el anterior (I_{ant}), y con respecto a un umbral establecido (U), calcula el movimiento de un objeto. Por lo tanto, dichos elementos deben eliminarse del fondo actual, de manera que ayude al módulo de detección de movimiento.

El algoritmo de la **Figuras 15 y 16** recorre cada uno de los *frames* de la secuencia de video y cada uno de los píxeles en las imágenes, al inicio el sistema se considera la primera imagen como fondo, después lo aprende a través de la gestión de intensidades por píxel, cuando el valor de permanencia (U_{est}) es

alcanzado o superado, el píxel de fondo (BG) es actualizado. Con dicho algoritmo se tendrá la seguridad de que la resta de imágenes para detectar los objetos en movimiento se procesará con un fondo que elimina las regiones homogéneas en las imágenes.

Los elementos más importantes del algoritmo que actualiza la imagen de fondo son el umbral (U1) y el U_{est}, ya que de éstos dos depende la eficiencia del proceso. A continuación se describe la función y el valor de dichos parámetros:

- **U1.** Es el umbral con el cual es comparado cada píxel de las imágenes, por lo tanto, cada diferencia entre un píxel de la imagen actual y la imagen anterior en la misma posición que supere a U1 se considera como un movimiento, lo cual quiere decir que el píxel no formará parte del fondo. Con base a lo expresado anteriormente, se puede decir que U1 depende de la intensidad de las imágenes y por consecuente de la calidad de las mismas
- $\boldsymbol{U_{est}}$**.** Como ya se había mencionado anteriormente, es el valor de permanencia de un píxel, este parámetro será comparado con el número de veces en el que un píxel tiene el mismo valor en la diferencia de la imagen actual y la imagen anterior, al ser alcanzado o superado, se determina que es estático y que debe formar parte de la imagen de fondo. Con base a lo expresado anteriormente, se puede decir que $\mathrm{U_{est}}$ depende de la velocidad en la que un píxel cambia de valor.

3.6.3 Diferencia de imágenes.

Ya calibrado el proceso que actualiza la imagen de fondo, se puede realizar la resta de imágenes para obtener cada uno de los objetos que se mueven. Por lo tanto, después de haber obtenido la diferencia absoluta entre imágenes, se procede a *umbralizar* los *frames* resultantes, esto con el objetivo de mejorar la intensidad de los píxeles en los objetos de *reflectancia* que se parecen a los de la pista de experimentación.

3.7 Seguimiento.

Ya habiendo aislado los objetos de interés se requiere entender su movimiento, el algoritmo propuesto debe ser capaz de localizar los vehículos de un *frame* durante la secuencia de video.

Existe una gran variedad de técnicas para seguir objetos en movimiento, las cuales se pueden clasificar como lo propone *Yilmaz et al. (2006)*:

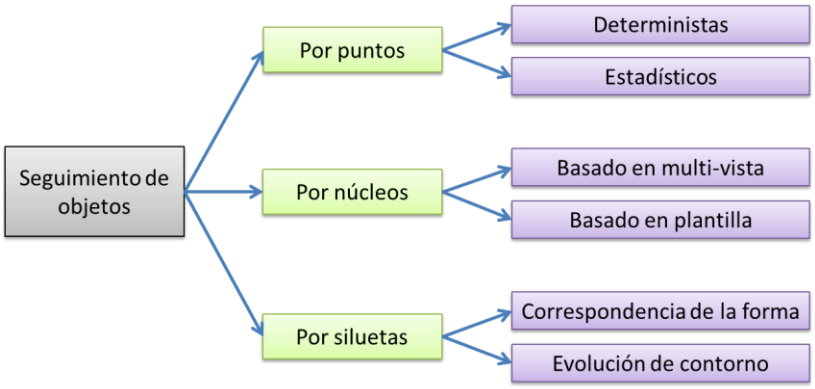


Figura 3-6 Taxonomía de métodos de seguimiento.

A continuación se describen las técnicas expuestas en la **Figura 3-6**:

- **Técnicas de seguimiento por puntos.** Esta técnica consiste en representar a cada uno de los objetos a través de uno o más puntos con características únicas que se pueden asociar con el estado del objeto en el *frame* anterior, calculando de esta forma su movimiento y posición. En este método deben elegirse puntos parametrizables (configurable o adaptable) con el objetivo de ser encontrados en otras posiciones de la imagen. Las técnicas de seguimiento de puntos se pueden clasificar en:
    - **Métodos deterministas.** Un método determinista asigna un costo de asociación de cada uno de los objetos en el *frame* $t-1$ a un único objeto en el *frame* t, la minimización del costo de correspondencia se considera como un problema de optimización que puede resolverse

a través de una correspondencia de uno a uno. Según Yilmaz et al. (2006) el costo de correspondencia generalmente se define usando una combinación de las siguientes restricciones:

- Proximidad. Asume que la posición de un objeto no cambia drásticamente de un *frame* a otro.
- Velocidad máxima. Define un límite de velocidad y limita la posibilidad de correspondencias a una zona circular del objeto.
- Cambios pequeños de velocidad. Asume que ni la velocidad ni la dirección de un objeto cambia drásticamente.
- Movimiento común. Limita la velocidad de los objetos en un pequeño vecindario asumiendo que los movimientos son similares.
- Rigidez. Asume que los objetos en un mundo tridimensional son rígidos, lo cual quiere decir que los puntos en el elemento actual no tendrá cambios.
- Uniformidad por proximidad. Es la combinación de la proximidad y las pequeñas velocidades.

- **Métodos estadísticos.** Este tipo de técnicas se realizan a través de la estimación de los estados de los objetos que se están siguiendo (observaciones e incertidumbres), es necesario modelar las propiedades de posición, velocidad y aceleración de cada uno de los elementos que se desean seguir. Algunos métodos de este tipo son:
  - Filtro Kalman. Estima los estados de un sistema lineal a través de distribución Gaussiana, la etapa de predicción y la de corrección. El filtro Kalman es un conjunto de ecuaciones matemáticas que calculan de forma eficiente y recursiva el estado de un proceso que minimiza la media del error cuadrado (enfoque que mide la precisión de un modelo de pronóstico), ver ejemplo en Leal et al. (2010).
  - Filtro de partículas. Consiste en estimaciones secuenciales Monte-Carlo (procedimiento que analiza distribuciones de variables aleatorias). El principal objetivo de esta técnica es hacer uso de

funciones de densidad de probabilidad y su evolución en el tiempo, ver ejemplo en Gutiérrez (2010).

- Filtro de probabilidad conjunta de datos. Se basa en estimaciones bayesianas (probabilidades condicionales de eventos aleatorios) de la correspondencia entre las características detectadas por los sensores, esto quiere decir que esta técnica calcula la probabilidad de los lazos característicos a través de un conjunto de medidas entre los diversos objetos, ver ejemplo en Juric et al. (2010).
- Seguimiento de múltiples hipótesis. Consiste en mantener varias hipótesis de correspondencia para cada objeto en cada periodo de tiempo, la posición del objeto es el conjunto más probable de correspondencia, este algoritmo puede soportar nuevos objetos que entran al campo de visión. Cada hipótesis predice la posición de cada objeto en el siguiente *frame* con base a mediciones de distancia, lo cual genera una nueva hipótesis, ver ejemplo en Huang et al. (2008).

- **Técnicas de seguimiento basadas en el núcleo.** Esta técnica consiste en el cálculo de la traslación y rotación del núcleo isotrópico (las propiedades físicas son iguales en cualquier dirección) de un *frame* al siguiente o a través de densidades en el flujo óptico de bloques subsiguientes, cuyos núcleos pueden ser detectados por plantillas rectangulares o elipses con un histograma asociado. Estas técnicas dependen del tipo de objeto a seguir, del número de elementos y del método utilizado para detectar el movimiento, por lo tanto se divide en dos categorías:
  - **Basado en plantillas y modelos de apariencia basados en densidad.** Según Yilmaz et al. (2006), este tipo de modelos han sido ampliamente utilizados por su relativa simplicidad y bajo costo computacional, a continuación se describen los más destacados:
    - Correspondencia de plantilla (template matching), el cual genera búsquedas exhaustivas de similitud de región con la plantilla del

objeto a buscar, como por ejemplo la correlación cruzada, a continuación se expresa la función de Yilma et al. (2006) :

$$\arg max_{dx,dy} \frac{\sum_x \sum_y (O_t(x,y) \times I_w(x+dx, y+dy))}{\sqrt{\sum_x \sum_y O_t^2}\,(x,y)}$$

Donde $\mathrm{I_w}$ es la imagen actual, $\mathrm{O_t}$ es la plantilla del objeto definida en el *frame* anterior y $(\mathrm{dx}, \mathrm{dy})$ es la posición de la plantilla candidata. Para generar la plantilla generalmente se usa la intensidad o el histograma de color y el gradiente de la imagen. Es importante mencionar que este método es de alto costo computacional debido a las búsquedas exhaustivas, sin embargo es posible limitarlas al vecindario de la posición previa para reducir dicho costo.

La búsqueda de los objetos a seguir también se puede hacer mediante el análisis de la media de color de los píxeles en una región rectangular, por lo tanto el objeto se compara en ocho posiciones vecinas, esto quiere decir que la posición del objeto es determinada por la similitud entre el modelo del objeto y la hipótesis de la posición, el que proporcione mayor relación entre la media de estos dos elementos será la seleccionada.

- Desplazamiento de media (*Mean Shift*), consiste en un proceso iterativo de punto fijo (método que estima una posición que es mejorada por iteración hasta la convergencia) que se encamina a un máximo local, en cada iteración se calcula el gradiente con base a la densidad y aplicándolo varias veces desde diferentes puntos de partida es posible encontrar las zonas en las que hay mayor densidad, esto quiere decir que dicho algoritmo pretende encontrar la posición del objeto en los *frames* siguientes comparando el histograma del objeto original con el histograma de las regiones candidatas.

- **Basado en modelos de apariencia multi-vista.** Consiste en la generación de múltiples vistas de forma offline con el objetivo de abarcar la apariencia de los cambios drásticos en los objetos al cambiar de dirección, es importante mencionar que este es un proceso previo al sistema de seguimiento en tiempo real.

  La técnica *Support Vector Machine (SVM)* es un esquema de clasificación que consiste en conjunto de muestras que etiquetan las distintas clases en una nueva muestra, esto quiere decir que si estas nuevas muestras se ponen en correspondencia en función de proximidad a las del modelo o esquema, es posible clasificarlas. Para algoritmos de seguimiento se crean clases positivas que representan a los objetos a seguir y clases negativas que constituye a todo aquello que no se debe seguir, por lo regular la muestra negativa son las regiones de fondo que pueden ser confundidas con los objetos. Ver un ejemplo en Valverde (2011).

- **Técnicas de seguimiento basadas en siluetas.** Esta técnica consiste en estimar la región de los objetos en cada uno de los *frames* a través de la densidad de apariencia o mapas de bordes, por lo tanto el seguimiento de siluetas se puede realizar por correspondencia de forma o evolución de contorno, a continuación su descripción:
  - **Correspondencia de forma.** Esta técnica es parecida a la de correspondencia de plantilla, consiste en buscar la silueta de un objeto con su modelo asociado en el *frame* actual, esto quiere decir que se debe hacer el cálculo de similitud entre las imágenes de la secuencia.
  - **Seguimiento del contorno.** A diferencia de la anterior el contorno del objeto evoluciona, esto quiere decir que la correspondencia del objeto es calculado con una silueta anterior a la de la imagen actual, por lo tanto la silueta del objeto se actualiza con base a la región del elemento en el *frame* actual. Existen dos formas de aplicar esta técnica, desarrollar un modelo de estados que representan la forma y

las características de movimiento, y a través de métodos probabilísticos que calculan el gradiente.

Ver un ejemplo de este tipo de métodos en Hilario et al. (2000).

A pesar de que un algoritmo de seguimiento basado en puntos es la solución óptima debido a su capacidad de soportar oclusiones, es un método que requiere almacenar todas las posibles correspondencias para actualizar su modelo en los siguientes *fotogramas*, esto implicaría problemas en un sistema en tiempo real, alta complejidad y por consecuente un alto costo computacional.

El seguimiento basado en siluetas implica un gran costo computacional, aún más que la técnica basada en puntos. Como ya se mencionó antes este tipo de métodos está orientado a objetos que no pueden modelarse con formas geométricas primitivas, por lo tanto hace uso de siluetas complejas para ser usadas como patrón de búsqueda.

Se descarta el seguimiento basado en patrones y el basado en modelos multivista, ya que el primero presenta problemas con objetos que varían constantemente su forma y el segundo no es recomendable para sistemas en tiempo real que requiere una gran velocidad de procesamiento.

De acuerdo a lo dicho anteriormente se cree que el seguimiento basado en apariencia a través de la técnica *mean-shift* es el más adecuado para cumplir con los objetivos de este proyecto, según Yilmaz et al. (2006), este tipo de modelos han sido ampliamente utilizados por su simplicidad y bajo costo computacional, además de que Garzón (2010), afirma que esta técnica es capaz de hacer el seguimiento de objetos que cambian de forma independientemente de la dirección y velocidad, es un modelo sencillo y general, con bajo costo computacional y robusto en cambios de iluminación, oclusiones e interacción con distintos objetos.

Es importante mencionar que la debilidad de *mean-shift* es que soporta sólo los solapamientos (objetos fusionados) que sucedan de un *frame* a otro, de lo contrario se pierde el seguimiento. Sin embargo, Garzón (2010) demuestra que es posible mejorar el algoritmo ante estas situaciones a través de una técnica de *ensemble traking*, el cual consiste en tratar el seguimiento de objetos como un problema de clasificación binaria, por lo tanto cada clasificador es aprendido con base a un conjunto de ejemplos positivos y negativos que puede ser combinado con un algoritmo *AdaBoost* (elemento que a través de aprendices débiles propone hipótesis adecuadas) para proporcionar mayor robustez al sistema.

Mean-Shift busca encontrar los picos o modos que representan los núcleos o valores de cada uno de los segmentos, contar, agrupar y etiquetar a través de los mismos. El algoritmo deberá buscar el *kernel* en los *frames* siguientes, en este caso es posible usar la intensidad de cada uno de los objetos, tal vez a través de sus histogramas, ya que se usará la computación acumulativa para segmentar. A continuación en la **Figura 3-7** se presentan las principales características de la técnica:

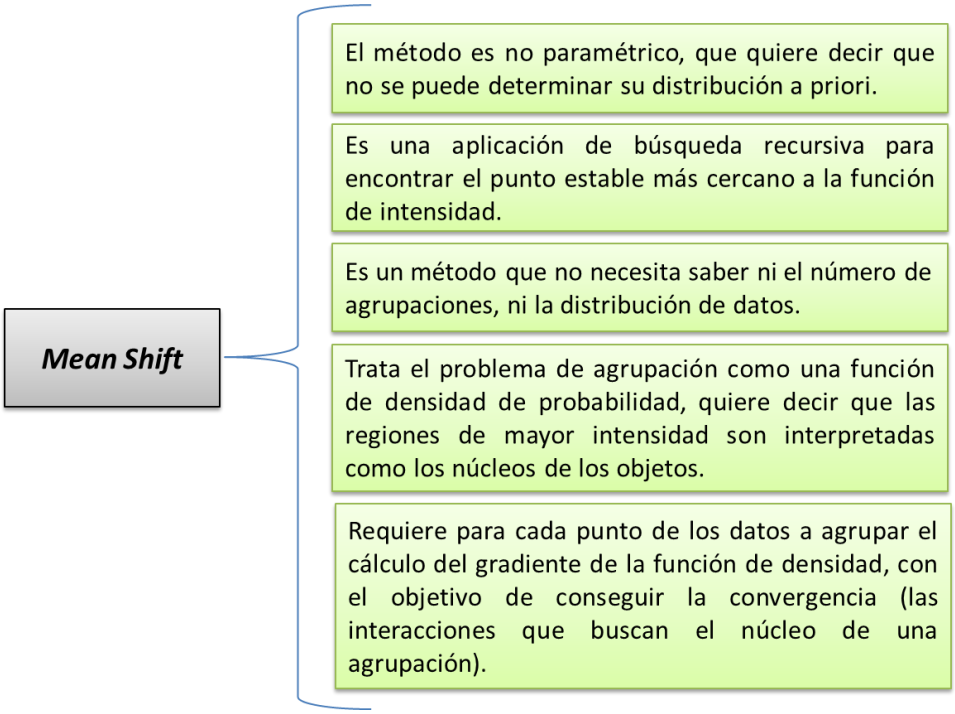


Figura 3-7 Principales características de *Mean Shift*.

A continuación se describen los pasos a seguir para desarrollar la técnica de seguimiento propuesta en esta tesis:

- **Estimar la función de densidad para obtener los máximos locales, la moda en las regiones densas y los valores bajos en regiones poco pobladas.** Según Martín (2011), la extracción de características de una muestra puede clasificarse en métodos paramétricos y no paramétricos. La principal diferencia entre estos dos grandes grupos, es que el segundo no requiere saber la suposición de la distribución de valores en el espacio. Por lo tanto, y a pesar de que los métodos no paramétricos no tienen la misma capacidad de interpretar los datos de forma exacta como los métodos paramétricos, es imprescindible hacer uso de ellos, ya que el objetivo de esta tesis es diseñar un clasificador que no requiera de información a priori de la muestra.

  Las técnicas más destacadas en la bibliografía para estimar la función de densidad según Martín (2011), son las *Ventanas de Parzen* y *K-vecinos más cercanos*, sus principales diferencia son:

  - Las *Ventanas de Parzen* realizan su estimación a través de un núcleo fijo, mientras que *K-vecinos más cercanos* se basa en un núcleo de ancho variable que depende de la densidad de la muestra.
  - Los resultados de *K-vecinos más cercanos* estima la función de densidad y clasifica los objetos al mismo tiempo, las *Ventanas de Parzen* estiman una función de densidad que después requerirá otro proceso de clasificación.

  El algoritmo de *K- vecinos más cercano* consiste en elegir un número de vecinos (en el caso K=4), elegir las métricas para calcular la distancia entre las muestras, calcular para cada muestra la distancia al resto de ellas y seleccionar los vecinos más cercanos. Para resolver empates de dos o más clases, se escoge la que tiene mayor probabilidad a priori, y si las probabilidades a priori coinciden, se elige al azar. Es importante mencionar que sólo se intenta ejemplificar la técnica a través de la imagen, ya que las métricas

para calcular distancias y las reglas de probabilidad de elección de clases dependen del mundo real del estudio.

A continuación en la **Figura 3-8** se presenta una imagen de la forma en la que la técnica de *K-vecinos más cercanos* estima la función de densidad:

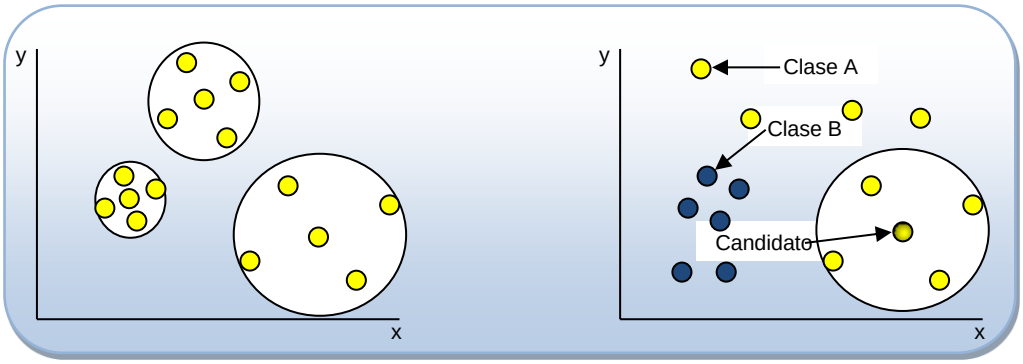


Figura 3-8 Una representación de *K-vecinos más cercanos*.

La técnica de *Ventanas de Parzen* consiste en el cálculo de probabilidad de pertenencia entre las muestras y una región definida por un hipercubo con un núcleo, por lo tanto el número de muestras que caen en dicha región estima los picos del diagrama de densidad.

A continuación en la **Figura 3-9** se presenta una imagen de la forma en la que la técnica de *Ventanas de Parzen* estima la función de densidad:

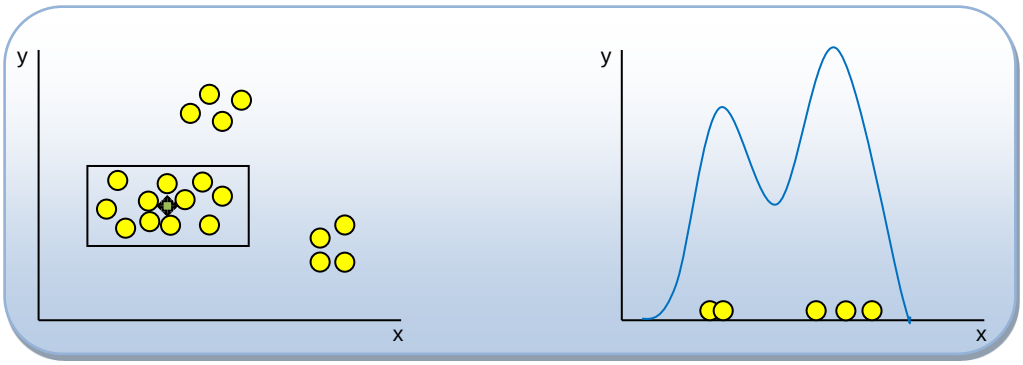


Figura 3-9 Una representación de *Ventanas de Parzen*.

La primera gráfica de la **Figura 3-9** ejemplifica el grado de pertenencia entre una ventana de volumen determinado y el conjunto de datos distribuidos, por otro lado, en la segunda gráfica se representa el histograma de una muestra con un tamaño equivalente a cinco y un ancho de ventana igual a dos (este tipo de valores se describen en el siguiente apartado de este documento).

Con base a lo descrito anteriormente se decide usar la técnica no paramétrica *Ventanas de Parzen*, ya que según Martín (2011), el proceso natural de *Mean Shift* es a través del método de estimaciones basadas en *Kernels*, el cual es mejor conocido en la literatura como *Ventanas de Parzen*.
Es importante mencionar que a través de esta técnica también es posible desarrollar ventanas *Gaussianas* que agreguen robustez contra el ruido y la ausencia de información entre la distribución de datos, conocer la dirección de las muestras es esencial para la discriminación de objetos.

Para poder estimar la función de densidad de una imagen es necesario primero entender su histograma, el cual representa el resumen estadístico de la distribución de los píxeles. Un ejemplo sencillo se puede visualizar en la **Figura 3-10** que a continuación se presenta:

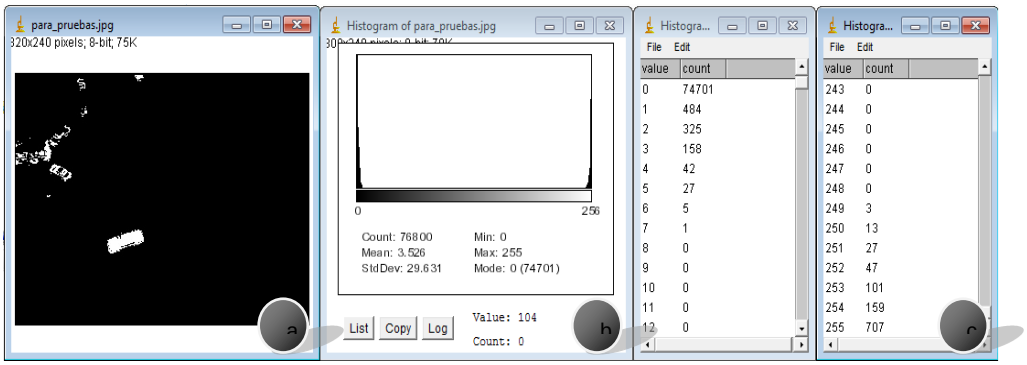


Figura 3-10 Representación de un histograma. (a) La imagen (b) El histograma (c) Lista de valores**.**

En la **Figura 3-10** se puede observar la representación del histograma de una imagen a escala de grises, los valores del fragmento (c) de la figura son el número de veces en el que se repite el valor de un píxel, por ejemplo, hay 74,701 píxeles con valor 0 y 707 con valor 255. Por lo tanto, son éstos los valores los que permiten la creación del histograma del fragmento (b), aunque no es posible verlo a simple vista, este objeto tiene un conjunto de barras con una altura que es determinada por dichos valores y que juntas forman la denominada curva de densidad, la siguiente figura ejemplifica este hecho:

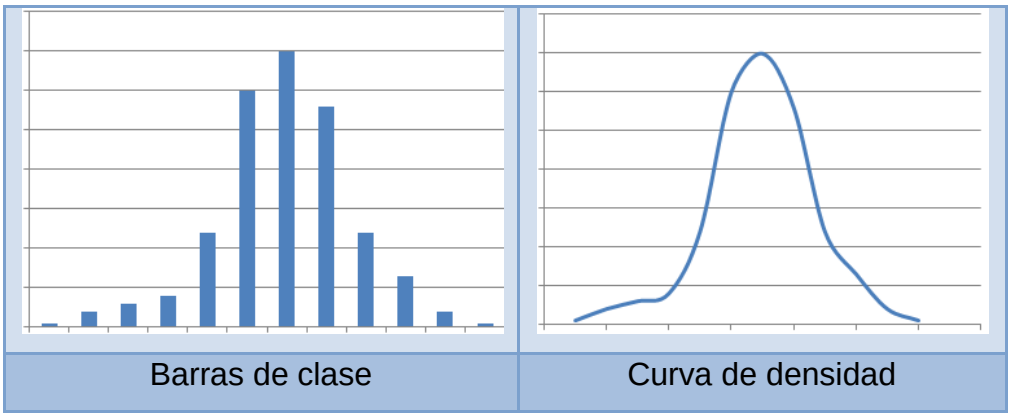


Figura 3-11 Distribución de valores.

A través del histograma de una imagen obtenemos la intensidad de cada uno de sus píxeles, la distribución de clases y la curva de densidad, según Servy et al. (2005), un histograma es un estimador de densidad valido, sin embargo no es suave, ni considera la correlación de datos de la distribución, esto quiere decir que no permite proporcionar una buena base para discriminar cada uno de los objetos de un *frame*.

Con base a lo descrito anteriormente, el método de *Ventanas de Parzen* según Servy et al. (2005), logra funciones de densidad suavizadas que construyen en cada punto de un eje real (valor de la muestra) un entorno denominado "Ventana", el cual se basa en dar prioridad a las muestras más cercanas a través de una función de ponderación de núcleo y así formar una curva de densidad como la presentada en la **Figura 3-11**. En resumen, se puede decir

que esta técnica tiene como principal objetivo aplicar al histograma un *Kernel* de suavizado que permita al vector propuesto por el método *Mean Shift* moverse sobre una función continua.

La *Función de Densidad de Probabilidad (FDP)* es el límite de un histograma cuando el ancho de cada subintervalo tiende a cero, esto quiere decir que es más fácil visualizar las probabilidades de los intervalos que calcular el valor exacto, por ejemplo, si el máximo local de un intervalo es 134, es posible estimar que se encuentra dentro del rango [133.5, 134.5].

Para poder hacer una estimación no paramétrica de la función de densidad, es necesario hacer uso de las variables aleatorias continuas, las cuales admiten cualquier número real y pueden tomar cualquier valor dentro de un intervalo predeterminado. Por lo tanto, el área bajo la curva de la función de densidad limitada por dos puntos representa la probabilidad de un valor, a lo cual se le denomina *Distribución de Probabilidad*.

Según Levin y Rubin (2010), una distribución de probabilidad representa las expectativas de que algo suceda y describe la forma en que se espera varíen los resultados. Una distribución de probabilidad discreta considera sólo un número limitado de valores mientras que una distribución de probabilidad continua puede tomar cualquier valor dentro de un intervalo dado. Por ejemplo, la probabilidad de que alguien haya nacido en un mes dado es discreta, ya que sólo existen doce posibles valores [1, 12], por otro lado, una distribución de probabilidad continua sería el precio medio del litro de gasolina para el próximo año, se estima que puede oscilar entre 10 y 14 pesos, por lo tanto el precio medio podría ser 10, 10.5, 11, 11.8, 12, 13, 13.343, 14, etc., es decir que existe una infinidad de posibilidades con la misma probabilidad.

Las variables aleatorias continuas permitirán asignar valores reales a intervalos predeterminados, sin embargo ahora es necesario calcular una función límite

de aproximación al histograma, esto significa que la probabilidad de un intervalo [a, b] será limitada por la función de densidad.

A continuación se presentan las propiedades de la función de densidad de probabilidad para variables aleatorias continuas Walpole (1999):

1. $f(x) \geq 0 \, para \, toda \, x \, \in R.$
2. $\int_{-\infty}^{\infty} \mathrm{f(x)} \, \mathrm{dx} = 1.$
3. $\mathrm{P}(a < X < b) = \int_{a}^{b} f(x)dx.$

Propiedad 1. Describe que la función de densidad siempre tiene que ser mayor o igual a cero para toda x perteneciente a la muestra, esto quiere decir que para que sea una función de densidad siempre debe ser positiva, recordemos que la probabilidad de un evento nunca puede ser negativa (regla básica de probabilidad).

Propiedad 2. La integral de toda la función de densidad debe ser igual a uno, esto quiere decir que el área total de probabilidad (de $-\infty$ a ∞) en una muestra tendrá un valor de uno.

Propiedad 3. La probabilidad de un evento es el resultado de la integral del cálculo del área bajo la curva de intensidad en un intervalo [a, b].

Ya que con una FDP es posible determinar la expectativa de que algo suceda, se puede poner como ejemplo el cálculo de la probabilidad de que un corredor tarde al menos dos horas en llegar a la meta, es necesario tomar en cuenta que un corredor puede terminar la carrera en cinco horas como máximo y una como mínimo.

A continuación en la siguiente figura se describe la solución del problema planteado anteriormente:

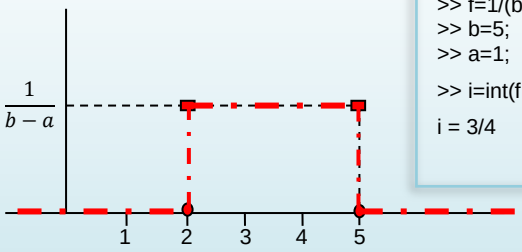
X = Es una variable aleatoria continua que representa el tiempo de llegada.

Nota: Es una distribución uniforme (significa que todos los valores dentro de un intervalo tienen la misma probabilidad), por lo tanto su función de densidad es:

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & para\ otro\ x \end{cases}$$

Código Matlab

```
>> syms x;
>> syms a;
>> syms b;
>> f=1/(b-a);
>> b=5;
>> a=1;
>> i=int(f,x,2,5)
i = 3/4
```



$$f(x) = \frac{1}{b-a} = \frac{1}{5-1} = \frac{1}{4},\ para\ 1 \leq x \leq 5$$

$$P(x \geq 2) = \int_2^5 \frac{1}{4} dx = \frac{3}{4} = 0.75$$

Por lo tanto la probabilidad de que un corredor llegue a la meta en al menos dos horas es del 75%.

Figura 3-12 FDP de una distribución uniforme.

En la **Figura 3-12** se describió un ejemplo de una distribución con valores que tienen la misma probabilidad, sin embargo antes de estudiar el método *Ventanas de Parzen* es necesario comprender aquellas distribuciones que a diferencia de las uniformes comprenden un campo de alta variabilidad, debido a esto se analizará la *Distribución Normal (o Gaussiana)*, ya que según Walpole (1999) es la más importante y está dotada de un conjunto de propiedades para modelar una gran cantidad de situaciones del mundo real.

La distribución de probabilidad de la variable normal depende de la media μ y su desviación estándar σ, sus principales características son:

- La curva normal tiene forma de campana y tiene sólo un pico en el centro, esto quiere decir que su distribución de probabilidad es simétrica debido a que la media, mediana y moda son iguales, además de que se localizan en el centro de la distribución (el pico).
- La curva de la normal desciende de forma suave en ambas direcciones (izquierda y derecha) a partir de su centro.
- Es asintótica porque la curva se acerca cada vez más al eje x pero nunca llega a tocarlo.

A continuación en la **Figura 3-13** se representa lo descrito anteriormente:

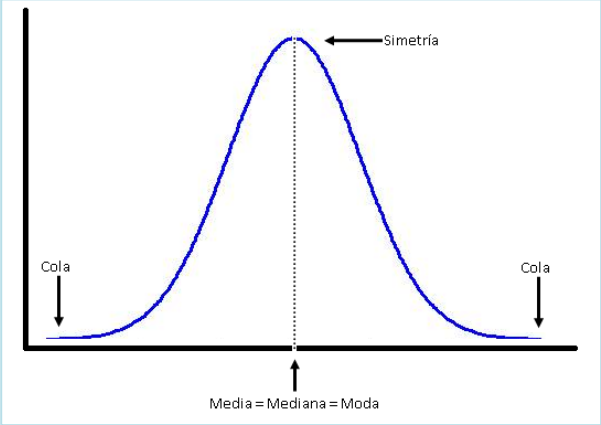


Figura 3-13 Representación de las características principales de una distribución normal.

De igual forma que las distribuciones uniformes, la probabilidad de que una variable aleatoria tome un valor determinado estará dada por el área entre dos números reales, es decir $\mathrm{P(a \leq X \leq b)} = \int_a^b \mathrm{f(x)dx}$, sin embargo su función de densidad se define de la siguiente forma:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

donde:

x, Es la variable con distribución normal

μ, Es la media o el valor esperado de *x*

σ, Es la desviación estándar o típica de *x*.

Según Díaz y Fernández (2001) se ha deducido que hay más de una distribución normal de forma similar a las otras que dependen de su media y su varianza, por lo tanto, proporcionar una tabla de probabilidades para cada combinación de μ y σ es imposible, para solventar el problema se usa la tabla de *Distribución Normal Estándar* cuya media es 0 y su desviación estándar es 1.

Con base a lo descrito anteriormente es necesario convertir una distribución real $\mathrm{X}\sim\mathrm{N}(\mu,\sigma)$ a una *Distribución Normal Estándar* $\mathrm{Z}\sim\mathrm{N}(0,1)$ con la expresión $\mathrm{Z}=\frac{\mathrm{X}-\mu}{\sigma}$, de esta forma tenemos que un valor Z mide la distancia entre un valor X y la media en las unidades de la desviación estándar para encontrar el área de probabilidad bajo cualquier curva normal haciendo referencia a la tabla del **Anexo 1** para valores Z positivos y **Anexo 2** para valores Z negativos. En la **Figura 3-14** se presentan algunos ejemplos.

Es importante mencionar que para los ejemplos de la **Figura 3-14** los valores de la variable aleatoria deben estar distribuidos normalmente, en caso contrario será necesario realizar la normalización o también denominada tipificación (según la fórmula expresada anteriormente), el resultado de dicho proceso generará una *Distribución Normal Estándar*. Esto se encuentra fundamentado en el *Teorema del Límite Central*, el cual afirma según Blaiotta y Delieutraz (2004), que la distribución de la suma de un número muy grande de variables aleatorias se aproxima a una distribución normal, esto quiere decir que al hacer un muestreo aleatorio suficientemente grande de una distribución asimétrica o no normal y calcular su media, resultará inevitablemente en una distribución normal o casi normal.

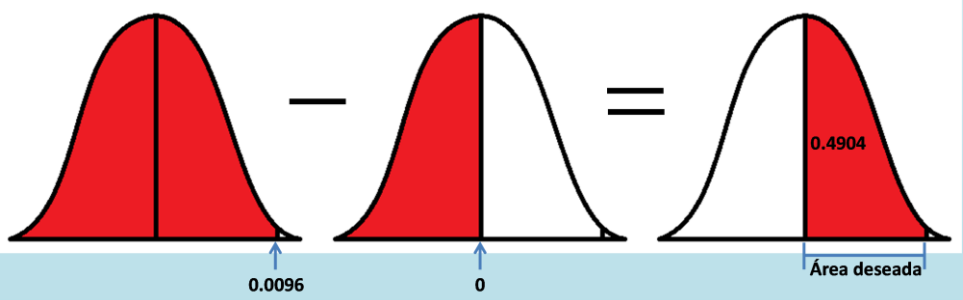
a) $P(Z < 2.34) = 0.9904$

z	.00	.01	.02	.03	.04
0.0	.5000	.5040	.5080	.5120	.5160
0.1	.5398	.5438	.5478	.5517	.5557
0.2	.5793	.5832	.5871	.5910	.5948
0.3	.6179	.6217	.6255	.6293	.6331
0.4	.6554	.6591	.6628	.6664	.6700
0.5	.6915	.6950	.6985	.7019	.7054
0.6	.7257	.7291	.7324	.7357	.7389
0.7	.7580	.7611	.7642	.7673	.7704
0.8	.7881	.7910	.7939	.7967	.7995
0.9	.8159	.8186	.8212	.8238	.8264
1.0	.8413	.8438	.8461	.8485	.8508
1.1	.8643	.8665	.8686	.8708	.8729
1.2	.8849	.8869	.8888	.8907	.8925
1.3	.9032	.9049	.9066	.9082	.9099
1.4	.9192	.9207	.9222	.9236	.9251
1.5	.9332	.9345	.9357	.9370	.9382
1.6	.9452	.9463	.9474	.9484	.9495
1.7	.9554	.9564	.9573	.9582	.9591
1.8	.9641	.9649	.9656	.9664	.9671
1.9	.9713	.9719	.9726	.9732	.9738
2.0	.9772	.9778	.9783	.9788	.9793
2.1	.9821	.9826	.9830	.9834	.9838
2.2	.9861	.9864	.9868	.9871	.9875
2.3	.9893	.9896	.9898	.9901	.9904

b) $P(Z > 2.34) = 1 - P(Z < 2.34) = 1 - 0.9904 = 0.0096$

El área total de probabilidad es igual a 1, por lo tanto y gracias a su simetría podemos restar el área total a la probabilidad que ya habíamos calculado.

c) $P(0 < Z < 2.34) = P(Z < 2.34) - P(Z < 0) = 0.9904 - 0.5000 = 0.4904$



Gracias a su simetría podemos restar dos áreas para encontrar la probabilidad de la diferencia.

Figura 3-14 Ejemplos de probabilidad de *Distribución Normal Estándar*.

La técnica que se usará para estimar la función de densidad en esta tesis, es la basada en *Kernels* ó también conocida en Visión artificial como *Ventanas de Parzen*, consiste en centrar hipercubos en cada una de las muestras de volumen y forma determinada para estimar la pertenecía que hay entre los datos, se considera como una estimación de tipo no paramétrica porque el

número de cubos y parámetros son proporcionales al tamaño de la muestra. A continuación en la **Figura 3-15** se puede ver su representación:

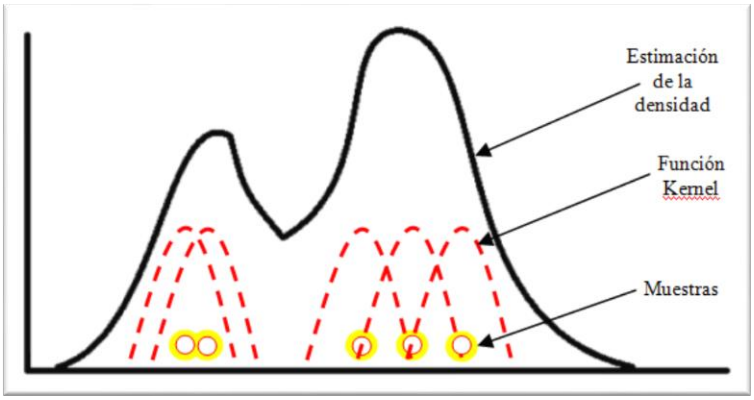


Figura 3-15 La representación de FDP basado en *Kernels*.

Las propiedades de las curvas de estimación en la técnica *Ventanas de Parzen* dependen de la función de núcleo (K) y el ancho de ventana (h), donde K determina la forma del pico y h el número de muestras que formarán parte de la ventana. Iglesias (2010) afirma que la forma del núcleo no es más importante que su ancho, sin embargo con la *Gaussiana* se puede aprovechar la matriz de covarianzas para obtener el grado de correlación entre las muestras, además del potencial en sus propiedades analíticas y de suavidad. Es por esto que en esta tesis se estimará la función de densidad a través de *Kernels Gaussianos*.

La técnica *Ventanas de Parzen* es expresada por los siguientes puntos:

A. La probabilidad de que una muestra x caiga dentro de la región R está definida por:

$$p_n(x) = \frac{k_n/n}{V_n}$$

donde,

k_n, Es el número de muestras que caen dentro de R.

V_n, Es el volumen de R.

n, Es el número total de datos.

Es importante mencionar que la fórmula citada en este punto, es la general para los métodos no paramétricos.

B. El volumen V_n del hipercubo R_n está definido por:

$$V_n = h_n^d$$

donde,

d, Es el número de dimensiones.

h_n, Es la longitud del hipercubo.

C. El número de datos que caen dentro del hipercubo está definido por:

$$k_n = \sum_{i=1}^{n} K\left(\frac{x - x_i}{h_n}\right)$$

donde,

n, Es el número total de datos.

x, Es el punto donde se ha centrado el hipercubo.

x_i, Es el elemento que podría estar dentro de dicho hipercubo.

h_n, Es el parámetro de suavizado o ancho del *Kernel*.

D. La función del *Kernel* o *Ventana de Parzen* está definida por:

$$K(u) = \begin{cases} 1 & |u_j| < \frac{1}{2} \\ 0 & en\ cualquier\ otro\ caso \end{cases}$$

La variable $K(u)$ define un hipercubo centrado en el origen, la función regresará 1 para todo x_i que se encuentra dentro del hipercubo centrado en x, y regresará 0 en caso contrario.

E. Sustituyendo la fórmula del cálculo del número de muestras que caen en el hipercubo y la fórmula general de probabilidad para métodos paramétricos ya descritas anteriormente da como resultado la estimación de la densidad mediante *Núcleos de Parzen*, su definición está dada por:

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h_n^d} K\left(\frac{x - x_i}{h_n}\right)$$

donde, la variable V_n es sustituida por h_n^d debido a que el volumen del hipercubo o *Kernel* está definido por su longitud elevado al número de dimensiones.

F. La solución de *García y Sancho (2010)* ante el problema de discontinuidad es el empleo de un *Kernel Gaussiano*, el cual está definido por:

$$K(u) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{1}{2}u^2\right)$$

G. Por lo tanto y con base a las definiciones anteriores, la función de densidad de probabilidad mediante *Ventanas de Parzen* y un *Kernel Gaussiano* está definido por:

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{(h_n\sqrt{2\pi})^d} exp\left(-\frac{1}{2}\left(\frac{x - x_i}{h_n}\right)^2\right)$$

A continuación en la **Figura 3-16** se describe la solución de un problema que busca encontrar la probabilidad de que $p_n[x = 4]$, dado el conjunto de datos $x_1 = 3,\ x_2 = 3.5,\ x_3 = 4,\ x_4 = 2$ y $x_5 = 7$, la función de densidad es estimada por una FDP de *Ventanas de Parzen*, un *Kernel Gaussiano* con $\sigma = 1$ y el parámetro de suavizado $h = h_5$:

Con la función del *Kernel* $K(u) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{1}{2}u^2\right)$ para cada muestra y la sustitución de σ:

$$\frac{1}{\sqrt{2\pi}} exp\left(-\frac{(x_1 - x)^2}{2}\right)$$

1) $K(x) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{(3-4)^2}{2}\right) = 0.2420$
2) $K(x) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{(3.5-4)^2}{2}\right) = 0.3521$
3) $K(x) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{(4-4)^2}{2}\right) = 0.3989$
4) $K(x) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{(2-4)^2}{2}\right) = 0.0540$
5) $K(x) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{(7-4)^2}{2}\right) = 0.0044$

Luego, como:

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n} K(x)$$

Entonces:

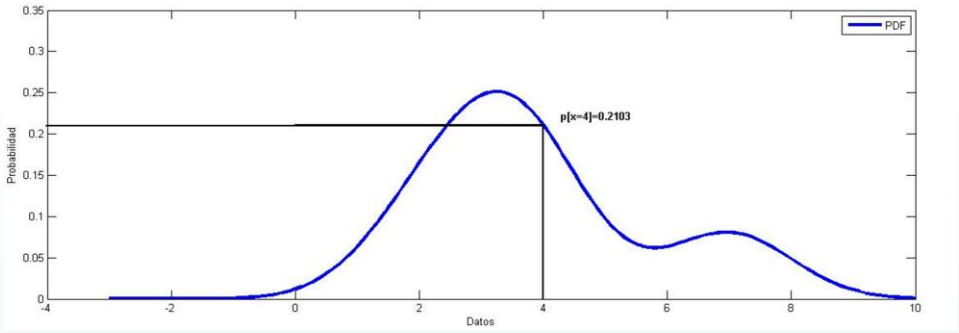
$$p_5[x = 4] = \frac{0.2420 + 0.3521 + 0.3989 + 0.0540 + 0 + 0040}{5} = 0.2103$$

Código Matlab

```
>> x=4; xn=[3 3.5 4 2 7]; px=0; n=1; h=1;
>> while n <= length(xn)
fx=1/(sqrt(2*pi))*exp(-0.5*((xn(n)-x)/h)^2);
px=px+fx;
if  n==length(xn)
px=px/length(xn);
end
n=n+1;
end
>> px
px = 0.2103
```

Figura 3-16 Ejemplo de FDP de *Ventanas de Parzen*.

La función de distribución de probabilidad del ejemplo anterior se muestra a continuación en la **Figura 3-17**:



Código Matlab de la función desarrollada

```
function p = Gaussian_ParzenWindow(m,vc,h)
%Determina la función de probabilidad en 1D
%   Las variables de entrada y salida son:
%   m  = El vector de la muestra
%   vc = Vector de espacio muestral
%   h = Volumen del hipercubo
%   p = Vector de probabilidad (salida)
z=1; p=zeros();
while z <= length(vc)
    n=1;
    px=0;
    kn=0;
    while n <= length(m)
    kn = kn + 1/(sqrt(2*pi))*exp(-0.5*((m(n)-vc(z))/h)^2);
    if n==length(m)
    px=(kn/length(m))/h;
    end
    n=n+1;
    end
p(z) = px;
z=z+1;
end
end
```

Invocación de la función en ejemplo planteado

```
>> m = [3 3.5 4 2 7];
>>x = [-3:0.1:10];
>>h = 1;
>>probabilidad=Gaussian_ParzenWindow(m,x,h);
>>plot(x,probabilidad);
```

Figura 3-17 Cálculo de FDP de *Ventanas de Parzen* en Matlab.

La altura de los picos en una FDP de *Ventanas de Parzen* representa el nivel de cercanía entre un dato y otro, esto quiere decir que un pico será tan alto como el número de muestras que caigan en el hipercubo $\mathrm{R_n}$, por otro lado, el parámetro de suavizado $\mathrm{h_n}$ representa el ancho de los picos, significa que si $\mathrm{h_n}$ es pequeño, únicamente las observaciones más cercanas serán relevantes. A continuación en la **Figura 3-18** se presenta un ejemplo:

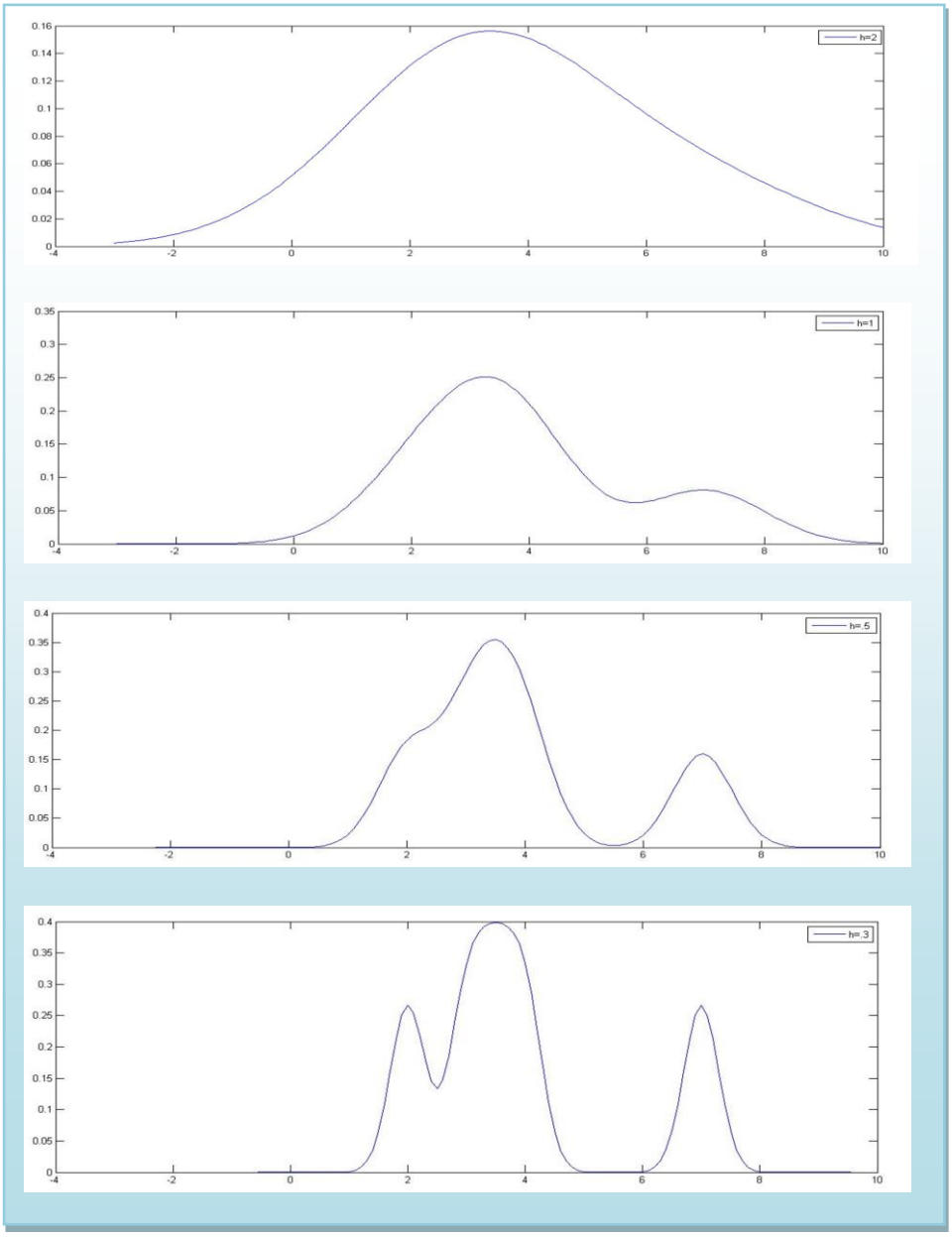


Figura 3-18 Representación del parámetro de suavizado h_n.

En la **Figura 3-18** se puede observar que el parámetro de suavizado $\mathrm{h_n}$ no sólo determina el ancho de los picos, sino también el número de ellos, por ejemplo, se observa que en el FDP con $\mathrm{h_n} = 2$ todas las muestras caen en un solo pico y al dar un valor de $\mathrm{h_n} = 0.3$ las observaciones se distribuyen en dos más.

Ya que se ha estudiado la funcionalidad FDP de *Ventanas de Parzen* y la importancia de los parámetros de función de núcleo (K) y ancho de ventana (h), es posible hacer uso de esta técnica para cumplir con el objetivo de la tesis, por tanto es necesario obtener la FDP de una imagen (considerada como una matriz), para lo cual, partiremos de la fórmula expresada por Wang et al. (2007) para estimar funciones de densidad por estimadores *kernel* en dos dimensiones, la cual se presenta a continuación:

$$p(x,y) = \frac{1}{2\pi N}\sum_{i=1}^{L}\sum_{j=1}^{C_i}\frac{1}{h_{C_i}^2}exp\left(-\frac{(x-x_j)^2(y-y_j)^2}{2h_{C_i}^2}\right).$$

donde,

$p(x,y)$, Es la probabilidad del valor localizado en el punto (x,y).

x, Es el eje de las abscisas coincidente con la línea horizontal de la matriz.

y, Es el eje de las ordenadas coincidente con la línea vertical de la matriz.

N, Es el tamaño o dimensión de la matriz representada por $(m\ x\ n)$.

L, Es conjunto de niveles de gris.

C_i, Es el número de puntos por nivel de gris.

h, Es el parámetro de suavizado o ancho del hipercubo.

x_j e y_j, Son los parámetros de localización de los puntos que podrían pertenecer al hipercubo.

A continuación se describen más detalladamente algunos de los elementos mencionados anteriormente que ayudarán a comprender mejor el procedimiento:

A. $N = m \times n$, entonces $x \in \{1,2 \dots, m\}$ e $y \in \{1,2 \dots, n\}$.

$$N = \begin{pmatrix} 11 & \cdots & 1n \\ \vdots & \ddots & \vdots \\ m1 & \cdots & mn \end{pmatrix}$$

B. L representa el número de niveles de gris en la matriz, por lo tanto, $i \in \{1,2 \dots, L\}$. Así, por ejemplo, si consideramos la siguiente matriz:

$N = \begin{pmatrix} 1 & 2 & 7 & 2 \\ 2 & 1 & 1 & 2 \\ 2 & 4 & 2 & 6 \\ 4 & 2 & 4 & 4 \end{pmatrix}$ Tendremos que $L = 5$ e $i = \{1,2,4,6,7\}$.

C. N estará conformado por $\sum_{i=1}^{L} C_i$. Por lo que en nuestro ejemplo tendremos que:

$\begin{pmatrix} 1 & 2 & 7 & 2 \\ 2 & 1 & 1 & 2 \\ 2 & 4 & 2 & 6 \\ 4 & 2 & 4 & 4 \end{pmatrix}$ Por lo tanto, $C_1 = 3, C_2 = 7, C_4 = 4, C_6 = 1, C_7 = 1$

D. x_j e y_j son los parámetros de localización de los puntos que podrían pertenecer al hipercubo, sin embargo, en vez de representar abscisa y ordenada como en el caso de x e y, los valores estarán definidos como $\omega_i = \{(x_1, y_1), (x_2, y_2) \dots, (x_{C_i}, y_{C_i})\}$,

lo cual quiere decir, que los conjuntos estará ordenados por niveles de gris ω_i.

$$\begin{pmatrix} 1 & 2 & 7 & 2 \\ 2 & 1 & 1 & 2 \\ 2 & 4 & 2 & 6 \\ 4 & 2 & 4 & 4 \end{pmatrix}$$ Por lo que en este caso:

$$\omega_1 = \{(1,1), (2,2), (2,3)\},\ \omega_6 = \{(3,4)\},$$

tomando en cuenta que $C_1 = 3$ y $C_6 = 1$.

Es importante dejar en claro que la función de Wang et al. (2007) descrita anteriormente, tiene como principal objetivo integrar la técnica de *Ventanas de Parzen* para estimar la distribución de probabilidad espacial de los valores de nivel de gris en una imagen, sin embargo para el propósito de esta tesis, se requiere que la FDP sea calculada con base a la cercanía de cada uno de los datos presentados en una muestra y no al valor del píxel.

Con lo anterior, se quiere decir que los picos Gaussianos generados por la FDP representarán a cada uno de los objetos en la imagen y serán detectados con base a los máximos locales expuestos por la probabilidad calculada. Cabe mencionar que el área de dichos elementos será la base del proceso de clasificación.

A continuación se describen las adecuaciones de la metodología de Wang et al. (2007) para calcular la FDP de una imagen y el conjunto de funciones requeridas para cumplir con el objetivo de esta tesis:

A. Función: Matriz de *convolución*.

   Una de las soluciones propuestas para cumplir con el objetivo de esta tesis es hacer uso de la operación local denominada *convolución*, Rodríguez y Sossa (2012) expresan que esta técnica produce mejores resultados que las operaciones puntuales y globales, lo cual se debe a varios factores: “El primero de ellos es

que permite realizar, localmente, un análisis más preciso de los niveles de grises y, entre otras cuestiones, estas operaciones toman en cuenta el comportamiento estadístico de los tonos de grises y la correlación existente entre ellos". Cabe mencionar que además de estas ventajas, la *convolución* ayuda a disminuir el tamaño de los conjuntos a procesar para calcular la probabilidad de un punto determinado, ya que ésta es calculada con base sólo a sus vecinos más cercanos. A continuación se describe la función que genera una matriz de *convolución* con base al valor de h:

Se asume que la matriz de *convolución* es un *Kernel* 2D de tamaño h, y que se posicionará su núcleo en cada uno de los datos de la muestra $(\mathrm{ren}, \mathrm{col})$, calcular dicho elemento se realiza de manera un tanto peculiar:

$$\mathrm{W} = \{(\mathrm{x}, \mathrm{y}) \in \mathbb{R}^2 : |\mathrm{x} - \mathrm{ren}| \leq \mathrm{h} \wedge |\mathrm{y} - \mathrm{col}| \leq \mathrm{h}\}$$

donde,

- ren, Es la posición renglón donde se encuentra el núcleo del *Kernel*
- col, Es la posición columna donde se encuentra el núcleo del Kernel
- h, Es el tamaño del Kernel
- x, Es la variable renglón del conjunto
- y, Es la variable columna del conjunto
- $\mathbb{R}^2$, Es el plano cartesiano donde las variables x, y toman cualquier valor real

Como podemos ver en la fórmula de la matriz de *convolución* se expresa un conjunto calculado con base a dos desigualdades de

valor absoluto, una para x y otra para y con la misma expresión, esto quiere decir que se ha determinado un rango de valores para la ordenada y la abscisa del mismo tamaño, así obteniendo como conjunto las posiciones vecinas con base a h y a la coordenada (x, y), a continuación se ejemplifica de forma gráfica un kernel de $\mathrm{h} = 1$:

(x,y)	1	2	3
1	(ren-1,col-1)	(ren-1,col)	(ren-1,col+1)
2	(ren,col-1)	(ren,col)	(ren,col+1)
3	(ren+1,col-1)	(ren+1,col)	(ren+1,col+1)

Figura 3-19 Matriz de *convolución* $h = 1$.

La cardinalidad del conjunto de *convolución* es expresada por $|W| = (2h + 1)^2$. Para que el *kernel* de suavizado funcione correctamente se requiere tomar en cuenta las siguientes restricciones al conjunto W:

1. $N_1 = \{(x, y) \in W : x \leq 0 \ \vee y \leq 0\}$
2. $N_2 = \{(x, y) \in W : x > m \ \vee y > n\}$
3. $N_3 = \{f(x, y) \leq 0\}$

Por lo tanto, el conjunto de *convolución* restringido será:

$$M_c = W - \{N_1 \cup N_2 \cup N_3\}$$

Las restricciones al conjunto W anulan aquellas coordenadas (x, y) que tengan valor 0 y/o estén fuera del rango del tamaño $\mathrm{m} \times \mathrm{n}$ de

la imagen. A continuación se muestra gráficamente un ejemplo de un *kernel* donde sus extremos quedan fuera de la imagen:

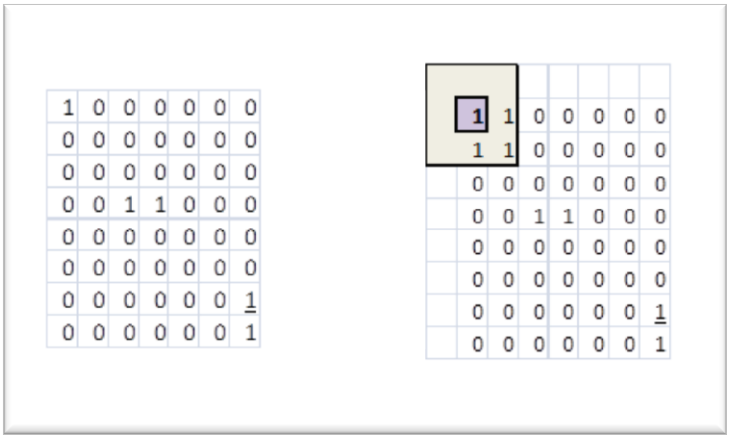


Figura 3-20 Ejemplo de un *kernel* de *convolución* $h = 1$ posicionando su núcleo en la coordenada de la imagen (1,1).

Como se puede observar en la **Figura 3-20**, el núcleo de la matriz de *convolución* se ha posicionado en la coordenada (1,1) de la imagen, por lo tanto tenemos posiciones que no deben ser tomadas en cuenta para el cálculo de la probabilidad, ésta es precisamente la tarea de los conjuntos de restricción, eliminar del conjunto de *convolución* final las posiciones con valor nulo, a continuación la comparación:

B. Conjunto con restricción.

$$M_c = \{(1,1), (1,2), (2,1), (2,2)\}$$

C. Conjunto sin restricción.

$$W = \{(0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2)\}$$

A continuación en la **Figura 3-21** se describe en código Matlab el procedimiento para calcular la matriz de *convolución*:

(x,y)	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	2	3	9	0	0
4	0	0	0	6	5	8	0	0
5	0	0	0	4	4	4	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

```
vr =
     3     3     3     4     4     4     5     5     5

vc =
     4     5     6     4     5     6     4     5     6

>> (3,4),(3,5),(3,6),(4,4),(4,5),(4,6),(5,4),(5,5),(5,6)
```

Código Matlab del cálculo de la matriz de convolución

```
function [vr,vc] = MatrizConvolucion(ren,col,m,n,muestra,h)
% ren   = renglón posición
% col   = columna posición
% m     = número total de renglones
% n     = número total de columnas
% vr    = vector renglón conjunto convolución
% vc    = vector columna conjunto convolución
% h     = Tamaño kernel (es en ancho de la matriz de convolución)
 incr=h*(-1);
inc=1;
while incr <= h
    incc=h*(-1);
    while incc <= h
    posx=ren+incr;
    posy=col+incc;
 if(posx>0 && posy>0 && posx<=m && posy<=n && muestra(posx,posy)~=0)
    vr(inc)=posx;
    vc(inc)=posy;
    inc=inc+1;
    end
    incc=incc+1;
    end
incr=incr+1;
end
end
```

Invocación de la función en ejemplo planteado

```
>>[vr,vc] = MatrizConvolucion(4,5,7,8,m,1);
```

Figura 3-21 Código Matlab para el cálculo del *kernel*.

D. Función: *Kernel Gausiano*.

Para la función de núcleo (K), se hará uso de la expresada por Wang et al. (2007), pero con una ligera modificación que se apegará a la naturaleza del procedimiento de *convolución* descrito anteriormente, por lo tanto:

$$K(x,y) = \frac{1}{2\pi} exp\left(-\frac{(x-x_j)^2+(y-y_j)^2}{2(2h+1)^2}\right)$$

donde,

x, Es la coordenada renglón donde se ha posicionado el núcleo de la matriz de convolución.

y, Es la coordenada columna donde se ha posicionado el núcleo de la matriz de *convolución*.

x_j, Es la coordenada renglón del conjunto calculado por la matriz de *convolución*.

y_j, Es la coordenada columna del conjunto calculado por la matriz de *convolución*.

h, Es el tamaño del *Kernel*.

A continuación, la comparación de la función de Wang et al. (2007) y la propuesta por esta tesis:

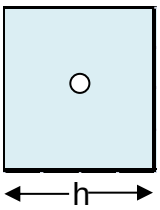
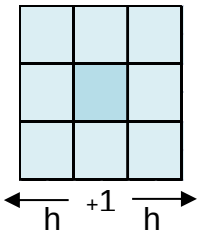
Wang et al. (2007): $K(x,y)=\frac{1}{2\pi}exp\left(-\frac{(x-x_j)^2+(y-y_j)^2}{2h_{C_i}^2}\right)$	Propuesta: $K(x,y)=\frac{1}{2\pi}exp\left(-\frac{(x-x_j)^2+(y-y_j)^2}{2(2h+1)^2}\right)$
1. La variable C_i denota un tamaño de *kernel* distinto para cada uno de los conjuntos, los cuales son las coordendas (x_j, y_j) de los niveles de gris presentados en la imagen.	1. Se ha eliminado la variable C_i, debido a que los conjuntos no son los distintos niveles de gris, sino los elementos calculados por una matriz de convolución que se basa en la cercanía de su núcleo.
2. La variable h^2 es el área del *kernel* expresada de la siguiente forma: 	2. Con la expresión $(2h+1)^2$ se incrementa el tamaño del *kernel* un reglón-columna, el cual es la posición donde se ha centrado el núcleo: 

Tabla 3-1 Diferencias en la función kernel propuesta.

E. Función: Función de densidad de probabilidad.

Habiendo definido las adecuaciones a la FDP de Wang et al. (2007), la fórmula propuesta por la tesis quedaría de la siguiente manera:

$$p(x,y) = \frac{1}{2\pi N}\sum_{j=1}^{M_c}\frac{1}{(2h+1)^2}exp\left(-\frac{(x-x_j)^2+(y-y_j)^2}{2(2h+1)^2}\right)$$

donde,

x Es la coordenada renglón donde se ha posicionado el núcleo de la matriz de *convolución*.

y, Es la coordenada columna donde se ha posicionado el núcleo de la matriz de convolución.

x_j, Es la coordenada renglón del conjunto calculado por la matriz de *convolución*.

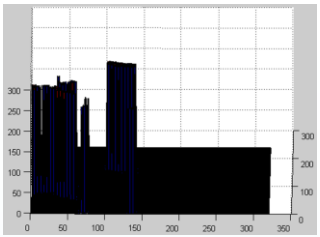
y_j, Es la coordenada columna del conjunto calculado por la matriz de *convolución*.

h, Es el tamaño del *Kernel*.

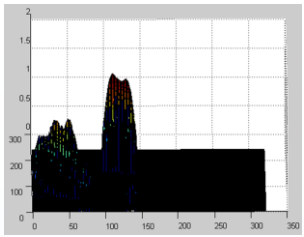
M_c, Es el conjunto (x_j, y_j) calculado por la matriz de convolución.

A continuación en la **Figura 3-22** se presenta un ejemplo de FDP en dos dimensiones y el procedimiento de programación realizado:

a) Imagen.



b) Vista superficie imagen original.



c) FDP con h=8.

Código Matlab del cálculo de la FDP

```
function mp = ConvolucionParzen(muestra,h)
% mp      = Matriz de probabilidad (salida) muestra   = Matriz de muestra (entrada)
% h       = Ancho de ventana (kernel de entrada)
[m,n]=size(muestra);    %m=renglones; n=columnas;
N=m*n; x=zeros(); y=zeros(); mp=zeros();
prob=0;
r=1;
   while r <= m
   c=1;
      while c <= n
      if(muestra(r,c)~= 0)
      [x,y] = MatrizConvolucion(r,c,m,n,muestra,h);  prob = CalculoConvolucion(r,c,x,y,h,N);
      mp(r,c)=prob;
      else
      mp(r,c)=0;
      end
      c=c+1;
      end
   r=r+1;
   end
end

function pd = CalculoConvolucion(rp,cp,vren,vcol,h,N)
%   rp   = Posición renglón   cp   = Posición columna   vren  = Vector renglón conjunto
%   vcol  = Vector columna conjunto   h    = Ancho de kernel   N    =Tamaño de muestra
inc=1; pd=0;
    while inc <= length(vren)
    xi=vren(inc); yi=vcol(inc);
    pd=pd+(1/((2*h+1)^2))*exp(-((rp-xi)^2+(cp-yi)^2)/(2*((2*h+1)^2)));
    inc=inc+1;
    end   pd=1/(2*pi*N)*pd; end
```

Invocación de la función en ejemplo planteado

```
>> img=imread('para_pruebas.jpg');
>> mp=ConvolucionParzen(img,8);
>> surf(mp)
```

Figura 3-22 FDP propuesta.

- **Encontrar los núcleos de los objetos**. Ya habiendo generado los picos en la imagen a través de la FDP con la técnica de *Ventanas de Parzen*, es posible detectar y hacer el conteo de los objetos encontrando los máximos locales. La mejor manera para encontrarlos es haciendo uso de una de las muchas aplicaciones de la derivada, en este caso, para determinar si una función $\mathrm{f(x)}$ es creciente o decreciente, por lo tanto, al encontrar un punto medio donde $\mathrm{f(x)}$ es creciente y decreciente en un determinado rango, podemos concluir que ese es uno de los picos *Gaussianos* generados por la FDP. Larry et al. (1990) expresa: "Se dice que la función $\mathrm{f(x)}$ es creciente en $\mathrm{x = a}$ si, en una pequeña región cercana al punto $\mathrm{(a, f(x))}$, la gráfica se levanta de izquierda a derecha. La función es decreciente en $\mathrm{x = a}$ si, en una pequeña región cercana al punto $\mathrm{(a, f(x))}$, la gráfica baja".

  Con base a lo anterior se puede decir que si los picos *Gaussianos* fueran perfectos o mejor dicho, sin píxeles internos con valor 0, un máximo local sería encontrado muy fácilmente, sólo es necesario encontrar la posición donde aplicando la derivada parcial a la **FDP** con *Ventanas de Parzen* expresada anteriormente, sea igual 0 en x e y. A continuación se describe la función:

$$\frac{\partial}{\partial x} f(x,y) = \frac{1}{2\pi N} \sum_{j=1}^{M_c} \frac{1}{(2h+1)^4} (-(x - x_j)) exp\left(-\frac{(x - x_j)^2 + (y - y_j)^2}{2(2h+1)^2} \right)$$

$$\frac{\partial}{\partial y} f(x,y) = \frac{1}{2\pi N} \sum_{j=1}^{M_c} \frac{1}{(2h+1)^4} (-(y - y_j)) exp\left(-\frac{(x - x_j)^2 + (y - y_j)^2}{2(2h+1)^2} \right)$$

  donde,

  x, Es la coordenada renglón donde se ha posicionado el núcleo de la matriz de *convolución*.

y, Es la coordenada columna donde se ha posicionado el núcleo de la matriz de *convolución*.

x_j, Es la coordenada renglón del conjunto calculado por la matriz de *convolución*.

y_j, Es la coordenada columna del conjunto calculado por la matriz de convolución.

h, Es el tamaño del *Kernel*.

M_c, Es el conjunto (x_j, y_j) calculado por la matriz de convolución.

Se ha expresado anteriormente que es posible encontrar máximos locales en una función determinando las pendientes de cada uno de los puntos, sin embargo, se ha comprobado que los picos *Gaussianos* generados presentan píxeles internos con valor 0, lo cual se debe a varios factores, entre ellos el ruido, defectos generados por los filtros de suavizado, cambios de luminosidad o calidad de captura en el sistema, por lo tanto no siempre se encontrará la horizontal ($\mathrm{f}'(\mathrm{x}) = 0$ y $\mathrm{f}'(\mathrm{y})$=0) en los picos de la escena. Por otro lado, también se concluye que no es viable determinar los máximos con base a la determinación de los puntos donde la función pasa de ser creciente a ser decreciente, esto es debido al mismo problema de calidad en los picos y a la naturaleza del procedimiento de *convolución*, ya que tendría que aplicarse un algoritmo más para seguir la forma de la función al pasar por cada uno de los picos, esto afectaría potencialmente el rendimiento de la solución propuesta en esta tesis.

Detectando las complicaciones expresadas anteriormente se ha decidido proponer una nueva función que calcule los máximos locales y que se apegue al procedimiento de *convolución*, esto con el objetivo de mejorar el tiempo de procesamiento y la obtención de resultados que también estén directamente

relacionados con sus procedimientos (la misma *Venta de Parzen* que genera la FDP, calcula los máximos locales).

Para reducir el tiempo de procesamiento, se reducirá la matriz de *convolución* de un cubo a una cruz, esto es posible gracias a la forma que la FDP le da a la escena y no hace necesario los valores que se encuentran en los extremos de la matriz, para esto es necesario modificar el conjunto de *convolución* de la siguiente forma:

$$\mathrm{r} = \{(\mathrm{x},\mathrm{y}) \in \mathbb{R}^2 : |\mathrm{x} - \mathrm{ren}| \leq \mathrm{h} \wedge \mathrm{y} = \mathrm{col}\ \}$$

$$\mathrm{c} = \{(\mathrm{x},\mathrm{y}) \in \mathbb{R}^2 : |\mathrm{y} - \mathrm{col}| \leq \mathrm{h}\ \wedge \mathrm{x} = \mathrm{ren}\ \}$$

$$\mathrm{M}_{\mathrm{cruz}} = \{\mathrm{r} \cup \mathrm{c}\}$$

donde,

- r, Conjunto renglón.
- *c,* Conjunto columna.
- ren, Posición renglón donde se encuentra el núcleo del *Kernel*
- col, Posición columna donde se encuentra el núcleo del *Kernel*
- h, Tamaño del *Kernel*
- x, Variable renglón del conjunto
- y, Variable columna del conjunto
- $\mathbb{R}^2$, Plano cartesiano donde x, y toman cualquier valor real
- $\mathrm{M}_{\mathrm{cruz}}$, Es la matriz resultante de la unión entre r y c.

Como podemos ver en la fórmula de la matriz de *convolución* anterior se expresa un conjunto para los renglones y otro para las columnas, con esto y al hacer la unión entre ellos, se logra encontrar sólo aquellas posiciones vecinas que forman parte de la horizontal y vertical del núcleo de la ventana $(\mathrm{ren}, \mathrm{col})$ y con base en h, a continuación se ejemplifica de forma gráfica un kernel de $\mathrm{h} = 1$:

(x,y)	1	2	3
1		(ren-1,col)	
2	(ren,col-1)	(ren,col)	(ren,col+1)
3		(ren+1,col)	

Figura 3-23 Matriz cruz de *convolución* $\mathrm{h} = 1$.

Para que el conjunto de *convolución* funcione correctamente, es necesario tomar en cuenta las mismas restricciones al conjunto W descrito anteriormente.

A continuación se describe la manera de obtener los máximos locales de la escena:

$$\mathrm{Ml_i} = \{(\mathrm{x}, \mathrm{y}) \in \mathrm{M_{cruz}} : \mathrm{f}(\mathrm{x}, \mathrm{y}) = \mathrm{Max}(\mathrm{M_{cruz}}) \wedge \mathrm{x} = \mathrm{ren} \wedge \mathrm{y} = \mathrm{col}\}$$

donde,

$\mathrm{ren},$	Posición renglón donde se encuentra el núcleo del *Kernel*
$\mathrm{col},$	Posición columna donde se encuentra el núcleo del *Kernel*
$\mathrm{x},$	Variable renglón del conjunto
$\mathrm{y},$	Variable columna del conjunto
$\mathrm{M_{cruz}},$	Es la matriz cruz de convolución
$\mathrm{Max},$	Es el valor máximo de la matriz cruz de *convolución*

Para solventar el problema de los píxeles internos a los picos con valor 0, se propone establecer una distancia euclidiana D que permita determinar el nivel de cercanía entre los máximos locales con el mismo valor en $f(x,y)$, a continuación se describe la solución:

A. $M_i = \{(x,y) \in Ml_i : f(x,y) = f(x_i,y_i), \forall\ (x_i,y_i) \in Ml_i\}$

B. $Ml_v = \{(x,y) \in M_{cruz}(x,y) \wedge (x_i,y_i) \in M_i : f(x,y) = f(x_i,y_i) \wedge \sqrt{(x-x_i)^2 + (y-y_i)^2} \leq D \wedge x \neq x_i \wedge y \neq y_i\}$

C. $Ml_m = M_{li} - Ml_v$

donde,

M_i, Es el conjunto de los valores $f(x,y)$ que se repiten en Ml_i

Ml_v, Es el conjunto de valores en M_i que se repiten y que tienen una distancia mínima entre ellas.

D, Es la distancia entre un máximo local y otro.

x, Variable renglón del conjunto M_{cruz}

y, Variable columna del conjunto M_{cruz}

x_i, Variable renglón del conjunto M_i

y_i, Variable columna del conjunto M_i

La solución consiste en crear un conjunto de máximos locales Ml_i, extraer de dicho conjunto aquellos valores que se repiten (conjunto del inciso A), calcular la distancia entre los elementos del conjunto M_i y el conjunto Ml_i, tales que tengan el mismo valor, sus coordenadas sean distintas y sea menor a D (conjunto del inciso B), y discriminarlos con base al conjunto Ml_i para obtener el conjunto final Ml_m.

Es importante mencionar que la distancia D debe ser calibrada posteriormente y que ésta dependerá del tamaño de la matriz de *convolución*.

La solución descrita anteriormente también permitirá establecer sólo una matriz de *convolución* para toda la clasificación vehicular, un matriz grande para un objeto más pequeño producirá una superficie como a continuación se presenta:

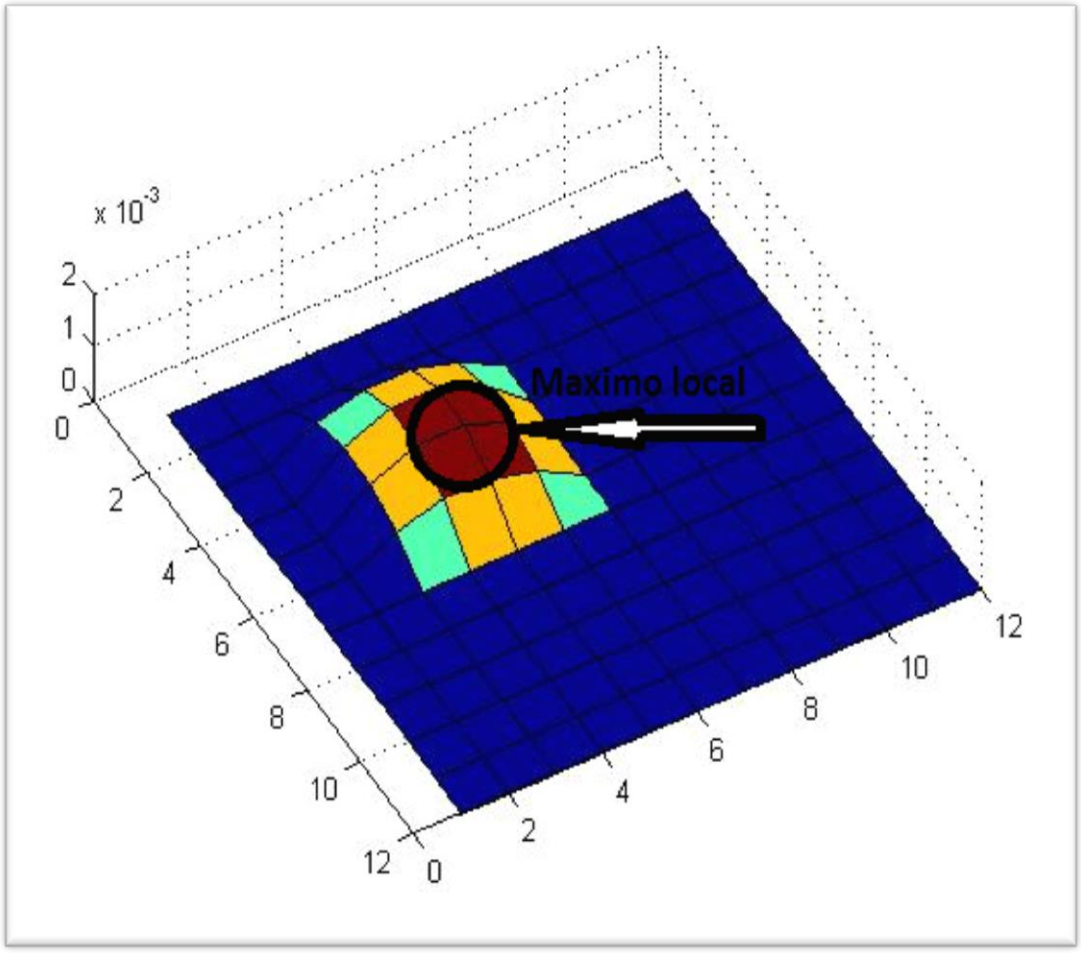


Figura 3-24 Superficie de una FDP de un objeto más pequeño que h.

Como se puede ver en la **Figura 3-24**, la matriz de *convolución*, en este caso de tamaño $\mathrm{h} = 1$, se generó un pico con cuatro píxeles máximos con el mismo valor, la solución $\mathrm{Ml_m}$ descrita anteriormente dio como resultado sólo uno, ya que a través de la distancia Euclidiana se pudo determinar que los cuatro píxeles del mismo valor eran el mismo máximo local.

A continuación se muestra el mismo caso experimental pero con píxeles internos con valor 0:

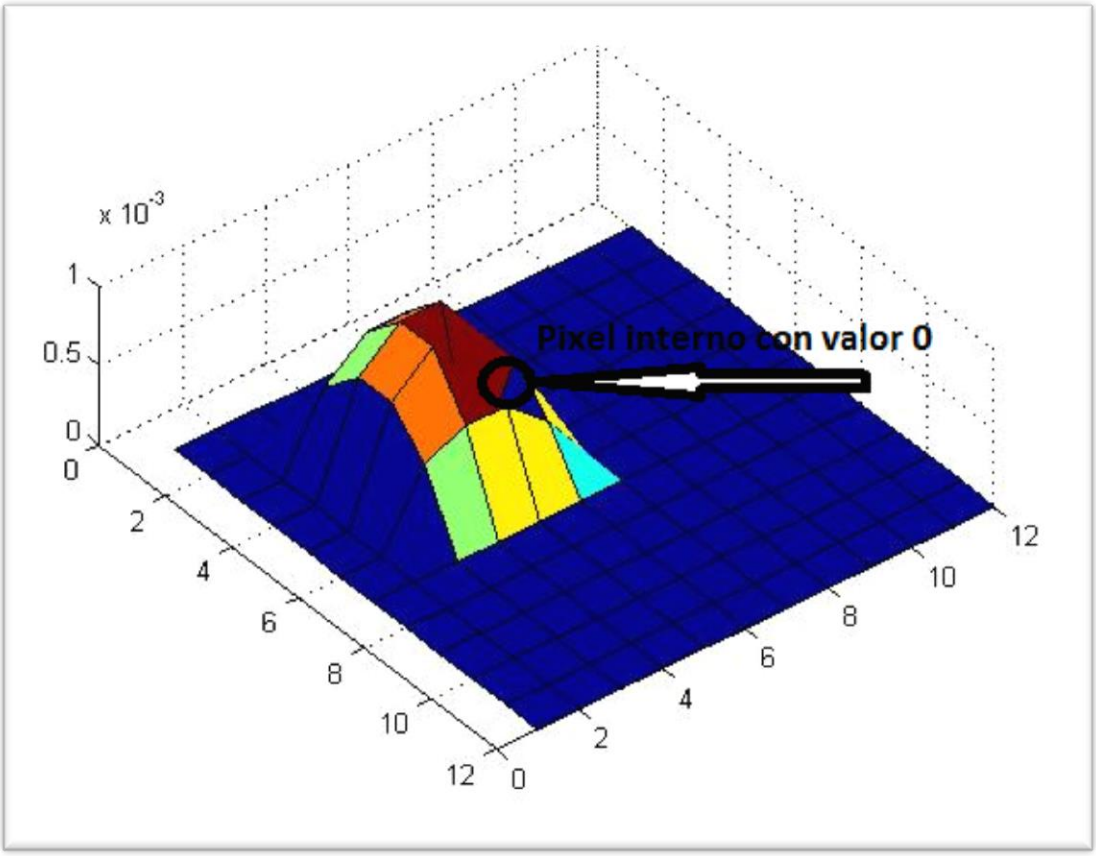
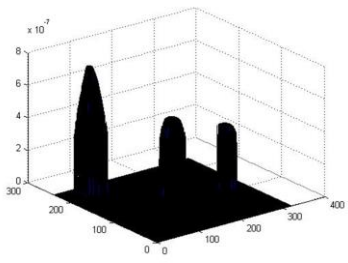


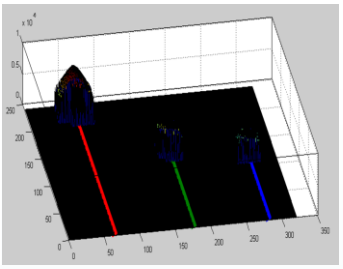
Figura 3-25 Superficie de una FDP con píxeles internos con valor 0.

La solución propuesta también logra atacar el problema de píxeles internos con valor 0, dando como resultado igualmente un máximo local pero ahora sólo de 3 valores.

Con base a la experiencia obtenida, la solución propuesta ha encontrado máximos locales en los extremos de los picos generados por la FDP, esto se debe a píxeles con valor 0 en las esquinas del objeto y su posición, sin embargo, será necesario establecer un umbral que elimine los máximos locales en el ruido causado por el sistema de captura de video, el nivel de luminosidad y los filtros de suavizado, por lo tanto, es este mismo umbral el que ayudará a discriminar máximos locales en las esquinas de los picos *Gaussianos*.

Por último, para finalizar con el proceso de detección de máximos locales, se presenta a continuación dos FDP extraídas del campo experimental:





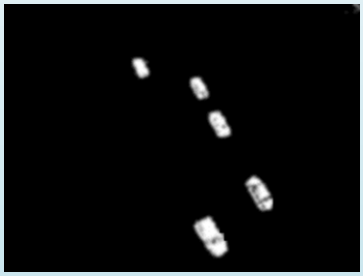
a) Imagen original **b) Surf FDP** **c) Máximos locales**

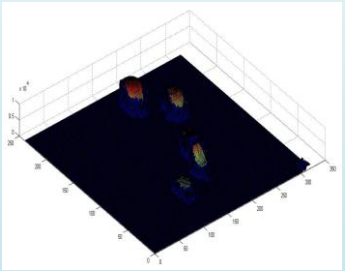
```
>> img=imread('image_matlab.jpg');
>> img=rgb2gray(img);
>> [mp,ml,mlx,mly,vcan] = ConvolucionParzen(img,25);

ml =

  1.0e-006 *

    0.3910    0.4801    0.7746

mlx =

   115   136   217

mly =

   281   176    65

vcan =

     1     2     3     0
```

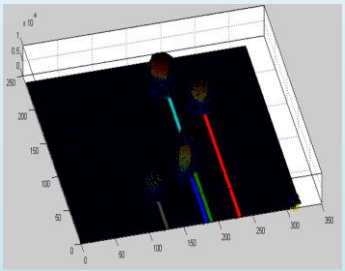
Descripción de variables.

- **ml** Valor máximo local
- **mlx** Posición renglón **ml**
- **mly** Posición columna **ml**
- **vcan** Discriminante de distancia

Nota: El discriminante de distancia expone las posiciones de los elementos discriminados o el conjunto del máximo local en un arreglo de un renglón.







a) Imagen original **b) Surf FDP** **c) Máximos locales**

```
>> [mp,ml,mlx,mly,vcan] = ConvolucionParzen(img,20);

ml =

  1.0e-005 *

    0.0163    0.0547    0.0802    0.0852    0.0953    0.1139

mlx =

     7    58    87    97   172   208

mly =

   313   124   181   188   229   188

vcan =

     1     2     3     4     5     6     0
```

Detección de ruido:

En este ejemplo se pude observar que el algoritmo detecto un máximo local donde hay ruido, por lo tanto hay que hacer una calibración para discriminar ese máximo local, en este caso sólo para ejemplificar, podría calibrarse con un umbral > a 0.02, ya que el ruido debería ser siempre los valores más pequeños, su valor resultante en este ejemplo es 0.0163 en las coordenadas (7,313).

Figura 3-26 Ejemplo de la solución propuesta en máximos locales.

- **Identificar y localizar los objetos de un *frame* a otro.** Hasta este punto se cuenta con los procesos de detección de objetos en movimiento, segmentación, función de densidad de probabilidad y localización de máximos locales, ahora para cumplir con el objetivo de esta tesis, es necesario desarrollar una función que localice a cada uno de los vehículos de un *frame* a otro hasta que salgan de la escena. El proceso de seguimiento requiere identificar cada uno de los elementos en el instante de tiempo t y el instante $t+1$, por lo tanto, la función es expresada por $f(x,y,t)$, siendo x, y las coordenadas espaciales de la escena. Es importante tomar en cuenta los posibles cambios que pueden ocurrir en los objetos, ya sea por el movimiento o por los cambios de luminosidad, con esto se quiere decir que es posible que los valores de color o estructurales en un vehículo cambien de un *frame* a otro.

  Gracias a las ventajas generadas por la aplicación de *Ventanas de Parzen* y la detección de cada uno de sus picos en cada uno de los fotogramas, podemos calcular las diferencias de varianzas entre los objetos más cercanos de un *frame* a otro y determinar su similitud y por consecuencia su movimiento.

  Existe una inmensa gama de métodos de seguimiento de objetos, entre los más populares están *Mean shift* y filtro de partículas, el primero busca de forma recursiva un punto estable más cercano a la función de densidad, creando un vector de media por cada punto encontrado, y al encontrar otro punto cercano estable calcula los histogramas para medir su similitud de distribución probabilística a través del coeficiente Bhattacharya, ver el método en Garrido (2008). El segundo utiliza un conjunto de partículas a las cuales se les asigna un peso de similitud y con la aplicación recursiva del método Monte Carlo estima la aproximación de un conjunto y otro, ver técnica en Gómez (2009).

  Como se puede observar los dos métodos para seguimiento de objetos mencionados son factibles, sin embargo, en esta tesis se busca una técnica que no requiera de procedimientos pesados que afecten el rendimiento del

sistema, por lo tanto se opta por un nuevo algoritmo que realice un análisis de varianza entre los picos definidos por los máximos locales cercanos al objeto a seguir, para así determinar el grado de similitud entre ellos. Con esto se conseguirá una función que no exija mucho tiempo de procesamiento y la mejora de algoritmos precedentes, también se pretende reducir líneas de código con la consecuente mejora en la velocidad de ejecución.

Según Levin et al. (2010) “Una técnica conocida como análisis de varianza (a menudo abreviada como ANOVA: analysis of variance), que permite probar la significancia de las diferencias entre más de dos medidas muestrales. Usando el análisis de varianza, podremos hacer inferencias acerca de si nuestras muestras se tomaron de poblaciones que tienen la misma medida.”.

El método de análisis de varianza utiliza la denominada lógica de la razón “F”, la cual según Sánchez (1992), “se fundamenta en el hecho de que se comparan dos estimadores de varianzas poblacionales, de forma que, si la hipótesis nula es cierta, ambos estimadores estiman la misma varianza”, dichos estimadores son medidas cuadráticas y sus componentes son las fuentes de variación estimada. Muchos autores manejan para facilitar la técnica ANOVA un formato denominado “Formato general de la tabla de análisis de la varianza”, a continuación en la **Tabla 3-2** se ejemplifica:

Fuente de variación	**Suma de cuadrados**	**Grados de libertad**	**Cuadrado Medio**
Entre los grupos	SST	$K-1$	$MSTR = \frac{SST}{K-1}$
En los grupos	SSE	$N-K$	$MSE = \frac{SSE}{N-K}$
Total			$\boldsymbol{F_{(MSTR/MSE)}}$

Tabla 3-2 Formato ANOVA.

A continuación se describe cada una de las variables expuestas en anterior:

- Suma de cuadrados entre los grupos.

$$SST = \left(\sum_{c=1}^{K}\left(\frac{t_c^2}{n_c}\right)\right) - \left[\left(\sum_{i=1}^{n_c} X_i\right)^2 / \left(\sum_{c=1}^{K} n_c\right)\right]$$

donde,

K, Es el número de muestras a evaluar.
t_c, Es la suma de los valores de cada muestra.
n_c, Es el número de elementos de cada muestra.
X_i, Es el valor de cada elemento de muestra.

- Suma de cuadrados en los grupos.

$$SSE = \left(\sum_{i=1}^{n_c} X_i^2\right) - \left(\sum_{c=1}^{K}\left(\frac{t_c^2}{n_c}\right)\right)$$

donde,

K, Es el número de muestras a evaluar.
t_c, Es la suma de los elementos de cada muestra.
n_c, Es el número de elementos de cada muestra.
X_i, Es el valor de cada elemento de muestra.

- Grados de libertad entre los grupos.

$$K - 1$$

donde,

K, Es el número de muestras a evaluar.

- Grados de libertad en los grupos.

$$N - K$$

donde,

K, Es el número de muestras a evaluar.

N, Es la suma de todos los n_c.

- Cuadrado medio entre los grupos.

$$MSTR = SST/(K - 1)$$

- Cuadrado medio en los grupos.

$$MSE = SSE/(N - K)$$

- Función de prueba.

$$F = MSTR/MSE$$

Una vez obtenida la función de prueba F, es necesario calcular el valor crítico con base en la distribución F, Según Levin et al. (2010) “Para llevar a cabo pruebas de hipótesis F debemos utilizar una tabla F, en la cual las columnas representan el número de grados de libertad del numerador y los renglones el número de grados de libertad del denominador. Existen tablas separadas para cada nivel de significancia”, por lo tanto, si el valor de la función de prueba excede el valor de la tabla, se rechaza la hipótesis nula y de lo contrario se aceptará.

A continuación en la **Figura 3-27** se presenta un ejemplo de cómo se extrae el punto crítico de una tabla de distribución F con un valor de significancia

$\propto = 0.01$, con grados de libertad entre los grupos $n_1 = 2$ y grados de libertad en los grupos $n_2 = 18$:

n_2 \ n_1	1	2	3	4	5	6	7	8	9
1	4052	5000	5403	5625	5764	5859	5928	5981	6022
2	98.50	99.00	99.17	99.25	99.30	99.33	99.36	99.37	99.39
3	34.12	30.82	29.46	28.71	28.24	27.91	27.67	27.49	27.35
4	21.20	18.00	16.69	15.98	15.52	15.21	14.98	14.80	14.66
5	16.26	13.27	12.06	11.39	10.97	10.67	10.46	10.29	10.16
6	13.75	10.92	9.780	9.148	8.746	8.466	8.260	8.102	7.976
7	12.25	9.547	8.451	7.847	7.460	7.191	6.993	6.840	6.719
8	11.26	8.649	7.591	7.006	6.632	6.371	6.178	6.029	5.911
9	10.56	8.022	6.992	6.422	6.057	5.802	5.613	5.467	5.351
10	10.04	7.559	6.552	5.994	5.636	5.386	5.200	5.057	4.942
11	9.646	7.206	6.217	5.668	5.316	5.069	4.886	4.744	4.632
12	9.330	6.927	5.953	5.412	5.064	4.821	4.640	4.499	4.388
13	9.074	6.701	5.739	5.205	4.862	4.620	4.441	4.302	4.191
14	8.862	6.515	5.564	5.035	4.695	4.456	4.278	4.140	4.030
15	8.683	6.359	5.417	4.893	4.556	4.318	4.142	4.004	3.895
16	8.531	6.226	5.292	4.773	4.437	4.202	4.026	3.890	3.780
17	8.400	6.112	5.185	4.669	4.336	4.102	3.927	3.791	3.682
18	8.285	6.013	5.092	4.579	4.248	4.015	3.841	3.705	3.597
19	8.185	5.926	5.010	4.500	4.171	3.939	3.765	3.631	3.523
20	8.096	5.849	4.938	4.431	4.103	3.871	3.699	3.564	3.457

Figura 3-27 Búsqueda del punto crítico en tabla de distribución F.

Por lo tanto, una vez obtenido el punto crítico $P_C = 6.013$ de la tabla de distribución F y una función prueba $F = 4.677$, se puede encontrar el intervalo o rango de aceptación de la hipótesis nula, como se ilustra en la **Figura 3-28**:

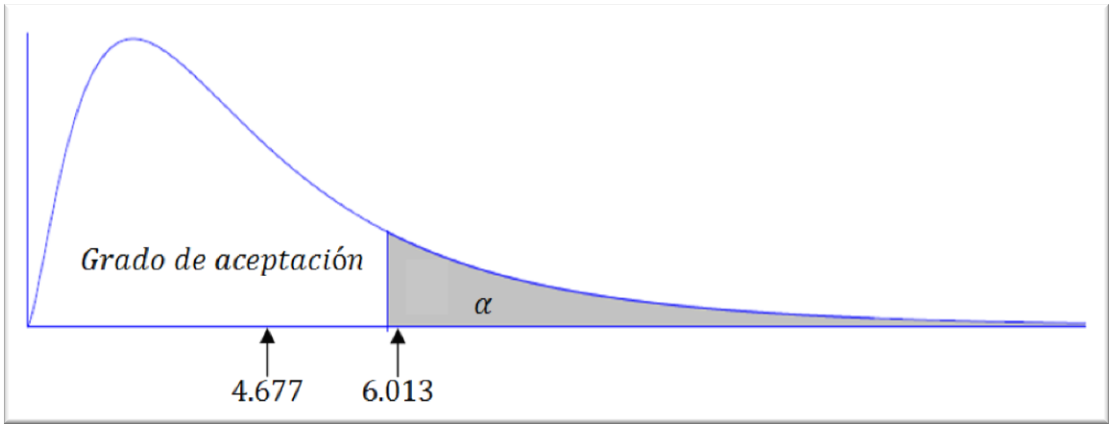


Figura 3-28 Representación del intervalo o rango de aceptación de una hipótesis.

A continuación en la **Figura 3-29** el código de programación de la función que determina el grado de similitud entre objetos:

Código Matlab de la función de similitud

```
function [ SST,SSE,K,NK,MSTR,MSE,F,PC ] = Similitud( muestras,alfa )
%   alfa        = Valor de significancia o de aceptación
%   muestras    = Las columnas hacen el cambio de muestra
%   K           = Grados de libertad entre las muestras
%   NK          = Grados de libertad dentro de las muestras
%   SST         = Suma de cuadrados entre los grupos
%   SSE         = Suma de cuadrados dentro los grupos
%   MSTR        = Cuadrado medio entre los grupos
%   MSE         = Cuadrado medio dentro de los grupos
%   F           = Función de prueba
%   PC          = Punto crítico extraído de la tabla F de probabilidad
[m,n]=size(muestras); %m=renglones; n=columnas;
r=1; tc=0; nc=0; cuadrados=0; tc_nc=0; SST=0; SSE=0; K=0; NK=0; MSTR=0; MSE=0;
while r <= m
c=1;
si=0;
nci=0;
    while c <= n
    if(muestras(r,c)~= 0)
    tc=tc+muestras(r,c);
    si=si+muestras(r,c);
    nc=nc+1;
    nci=nci+1;
    cuadrados=cuadrados+muestras(r,c)^2;
    end
    c=c+1;
    end
    tc_nc=tc_nc+((si^2)/nci);
r=r+1;
end
SST=tc_nc-(tc^2)/nc;
SSE=cuadrados-tc_nc;
K=m-1; NK=nc-m; MSTR=SST/K;
MSE=SSE/NK;
F=MSTR/MSE;
PC=finv(alfa,K,NK);%Es un valor invertido, esto sería .95 = 0.05
end
```

Invocación de la función en ejemplo planteado

```
>> m=[14 15 16 13 9 15 16;16 17 15 14 8 12 14; 13 15 14 17 13 10 8];
>> [sst,sse,k,nk,mstr,mse,f,pc]=Similitud(m,0.95)
```

Nota: Cada una de las muestras debe ser representada por un renglón de la matriz y la entra de la función *0.95* es un valor inverso, ya que se busca $\alpha = 0.05$.

Figura 3-29 Código de similitud entre objetos.

El objetivo de la función de similitud presentada en la **Figura 3-29** es determinar cuánto se parece un objeto en el tiempo t con los demás más cercanos en el tiempo $t+1$, de esta manera es posible determinar la localización de los vehículos en cada instante o en cada uno de los *fotogramas* de la secuencia de video.

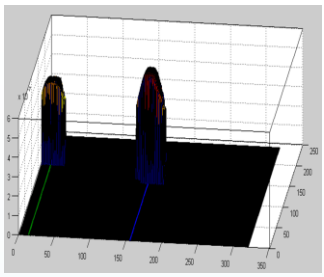
Cada *frame* es modificado por la función de *convolución Ventanas de Parzen*, esto quiere decir que los picos *Gaussianos* representados por cada vehículo tienen la misma varianza, por lo tanto, para calcular el grado similitud entre ellos, será necesario usar las imágenes en escala de gris.

Es importante mencionar que si se usarán muestras a color para esta técnica se obtendrían mejores resultados, ya que cada muestra cuenta con los tres factores RGB y las diferencias entre conjuntos serían más significativas.

La complicación al usar RGB es que requiere de más tiempo de procesamiento e incrementa el nivel de complejidad, por otro lado, con los conjuntos en nivel de gris y la distancia entre los objetos en distintos instantes de tiempo, se buscará solventar la desventaja resultante de compararlo con imágenes RGB.

A continuación en la **Figura 3-30** se presenta un ejemplo de los resultados generados por la función de similitud:

a) Imagen original *frame* 1.

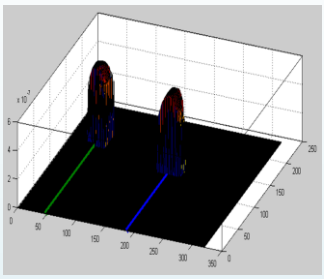


b) Surf FDP.

```
ml =
  1.0e-006 *
    0.5643     0.4230
mlx =
   151    180
mly =
   156     15
```

c) Localización de máximos locales.

a) Imagen original *frame* 1.



b) Surf FDP.

```
ml =
  1.0e-006 *
    0.5520     0.5295
mlx =
   140    171
mly =
   187     48
```

c) Localización de máximos locales.

Análisis de Varianza:

Núm.	HIPÓTESIS	*ANOVA* ($\alpha = 0.02$)	
		Función de prueba	Punto crítico
1	***Frame 1* vs *Frame 12***	$F = 5.6010$	$PC = 6.6744$
2	***Frame 1* vs *Frame 12***	$F = 0.0422$	$PC = 6.6744$
3	***Frame 1* vs *Frame 12***	$F = 29.2449$	$PC = 6.6744$
4	***Frame 1* vs *Frame 12***	$F = 2.2588e^{+003}$	$PC = 6.6744$

La hipótesis nula de esta prueba es:

$$H_0: \mu_1 = \mu_2$$

Por lo tanto, la hipótesis alternativa es:

$$H_1: \mu_1 \neq \mu_2$$

Lo cual quiere decir que cuando la función de prueba es menor al punto crítico, se asume que un vehículo es el mismo en el ***frame* 1** y el ***frame* 2**, se puede observar que el punto crítico en las cuatro hipótesis es el mismo, ya que la muestras tienen el mismo número de elementos y por consecuencia los mismo grados de libertad. Con base a lo descrito anteriormente se concluye que en este ejemplo H_0 se cumple en la hipótesis 1 y 2, y H_1 en la hipótesis 3 y 4.

Figura 3-30 Resultados de la función de Similitud.

Para finalizar con la técnica de seguimiento se requiere identificar cada uno de los vehículos y su localización hasta el momento en que desaparezcan de la escena, por lo tanto, es necesario tomar en cuenta los siguientes dos factores para la función de seguimiento de objetos en movimiento:

- **Elementos que provocan errores**. Son todos aquellos eventos y/o elementos que perjudican la secuencia del proceso de seguimiento, a continuación los más importantes:
  - **Las oclusiones y disoclusiones**. Una oclusión es un evento donde un objeto se encima o eclipsa en otro, la disoclusión es lo contrario, sucede cuando en la siguiente escena los objetos unidos aparecen separados. El problema de dichas acciones es que la secuencia de seguimiento en los objeto se perdería.
  - **El objeto deja de moverse por algunos instantes de tiempo**. Debido a la naturaleza del proceso de detección de objetos en movimiento (Computación Acumulativa), cuando un objeto se detiene, el sistema de visión deja de detectarlo, desaparece de la escena hasta que vuelva a moverse y por consecuencia es contabilizado nuevamente.
  - **Un objeto puede ser estadísticamente similar a otro**. La función de similitud tiene la responsabilidad de discriminar un vehículo en el instante de tiempo t y el instante $t+1$ de forma estadística, lo cual quiere decir, que dos objetos serán el mismo en distintos frames cuando su varianza de color sea muy parecida, sin embargo, cabe la posibilidad de que surja uno más con un grado de apariencia aceptable, en consecuencia el proceso perderá la secuencia de seguimiento de alguno de estos elementos.
- **Elementos que ayudan a reducir errores**. Son todos aquellos elementos que permiten solucionar los problemas en el proceso de seguimiento, a continuación los más importantes:
  - **Las fronteras**. Son parte de la escena y serán reconocidas como el acceso de entrada y salida de un vehículo, estos elementos son

parte de la calibración del sistema de visión desarrollado en esta tesis, lo cual quiere decir que deben ser localizadas a priori. El objetivo principal de las fronteras será determinar en qué momento un objeto aparece por primera vez y cuando sale de la escena.

- **La distancia entre *frames***. La distancia entre los máximos locales en el tiempo t permitirá discriminar aquellos objetos que pudieran tener un alto grado de similitud en el tiempo $t+1$, esto quiere decir, que dos objetos en distintos instantes no serán el mismo si la distancia es mayor a un umbral U_{dist}.
- **El desplazamiento**. El desplazamiento de un vehículo es uno de los factores más importantes para el control de oclusiones/disoclusiones, esta información permite conocer la dirección y línea temporal de localización, en consecuencia, es posible determinar en qué momento dos elementos son fusionados.
- **Latencia**. Según Márquez (2011) “es un coeficiente que refleja la continuidad de los objetos en cuadros consecutivos”, el parámetro estará normalizado y un valor muy cercano a 1 indicará que un vehículo ha aparecido de manera consecutiva durante la secuencia. La latencia junto con la información de desplazamiento y la localización de las fronteras, son los factores que permitirán determinar cuándo se presentan oclusiones/disoclusiones. Si en determinado instante de tiempo en dos o más objetos el coeficiente de latencia disminuye considerablemente y su dirección es muy parecida, es muy posible que haya ocurrido una oclusión, por otro lado, si la latencia disminuye y aparece un nuevo objeto lejos de las fronteras, es muy posible que haya ocurrido una disoclusión.

Con base a lo descrito anteriormente, el procedimiento de seguimiento se presenta a continuación:

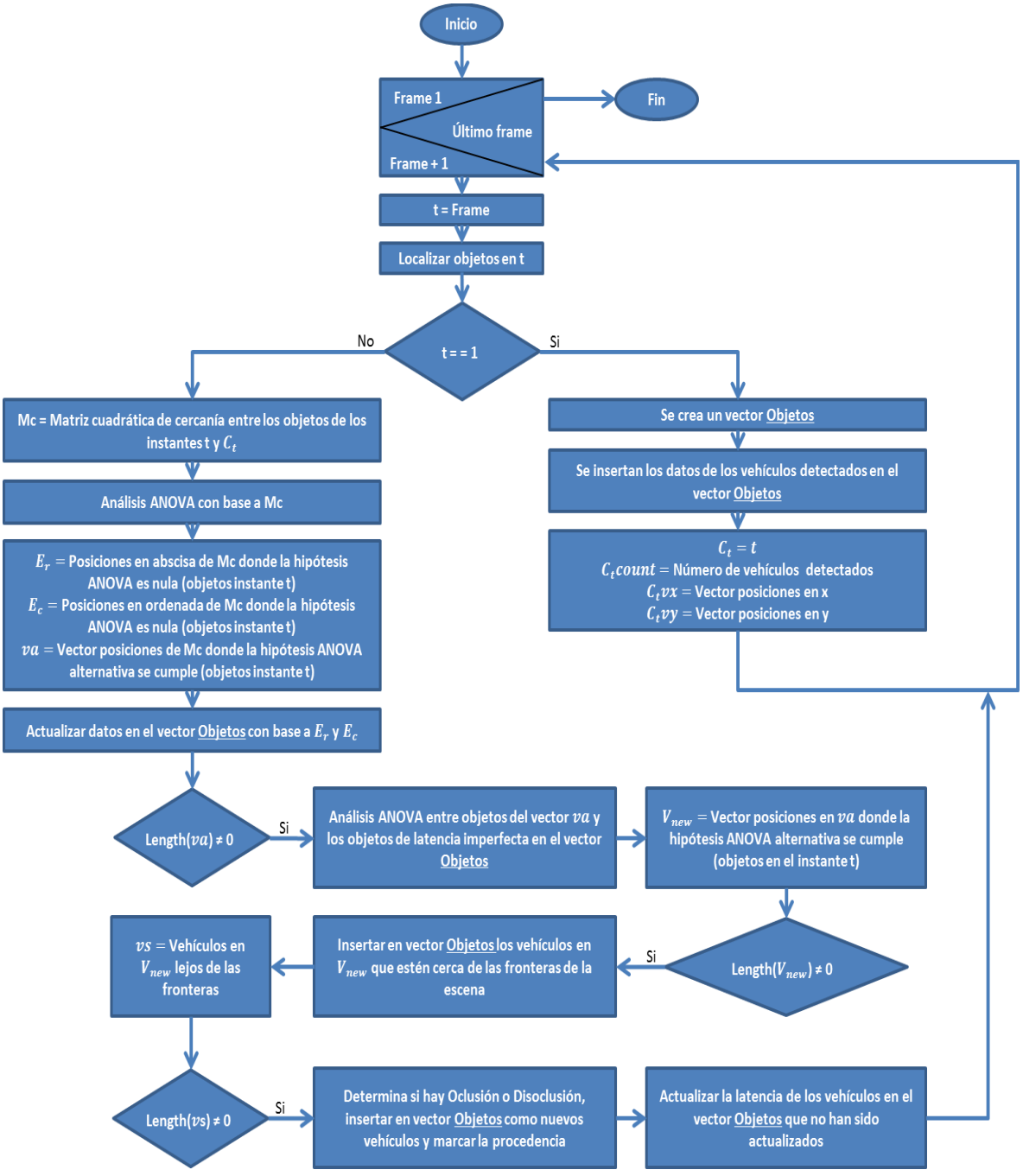


Figura 3-31 Algoritmo de seguimiento.

El algoritmo presentado en la **Figura 3-31** considera el riesgo de que los objetos en el instante $t+1$ puedan evadir el proceso de Similitud con ANOVA, los posibles motivos son:

1. Los niveles de luminosidad cambian drásticamente y por consecuencia, los niveles de color en el objeto se modifican.
2. El objeto acaba de entrar a la escena.
3. El objeto se ha fusionado con otro muy cercano (oclusión).
4. El objeto estaba fusionado y se ha separado (disoclusión).

Para estos casos, se ha incluido una serie de reglas que permiten discriminar cada uno de estas suposiciones, a continuación se describen:

1. Si el objeto no reconocido por el proceso de Similitud con ANOVA está lejos de las fronteras de la escena y si realizando otro análisis ANOVA con los objetos de baja latencia cumple con la hipótesis nula, significa que hubo un cambio drástico de luminosidad en la escena.
2. Si el objeto no reconocido por el proceso de Similitud con ANOVA está cerca de las fronteras de la escena, significa que es un vehículo nuevo.
3. Si las coordenadas del núcleo de un objeto no reconocido por el proceso de Similitud con ANOVA en el instante t forman parte de otro objeto tampoco reconocido por ANOVA en el instante $t+1$, ha sucedido una oclusión.
4. Si las coordenadas del núcleo de un objeto no reconocido por el proceso de Similitud con ANOVA en el instante $t+1$ forman parte de otro objeto tampoco reconocido por ANOVA en el instante t, ha sucedido una disoclusión.

Con base a los motivos de riesgo y las reglas de discriminación se hace indispensable el desarrollo de los procesos que lo controlen, a continuación se presentan los algoritmos de programación propuestos:

- Función **“Latencia imperfecta”.** Consiste en realizar un segundo análisis ANOVA para determinar el grado de similitud entre los vehículos de hipótesis

alternativa en el instante $t+1$ y aquellos vehículos de latencia baja que se encuentran en el historial de vehículos (vector Objetos), este algoritmo busca constantemente encontrar la similitud entre objetos no reconocidos, esto quiere decir, que si un objeto pierde por algunos instantes sus niveles de color originales o por lo menos los que tiene desde que entra a escena, existirá la posibilidad de ser reconocido por este procedimiento en instantes futuros:

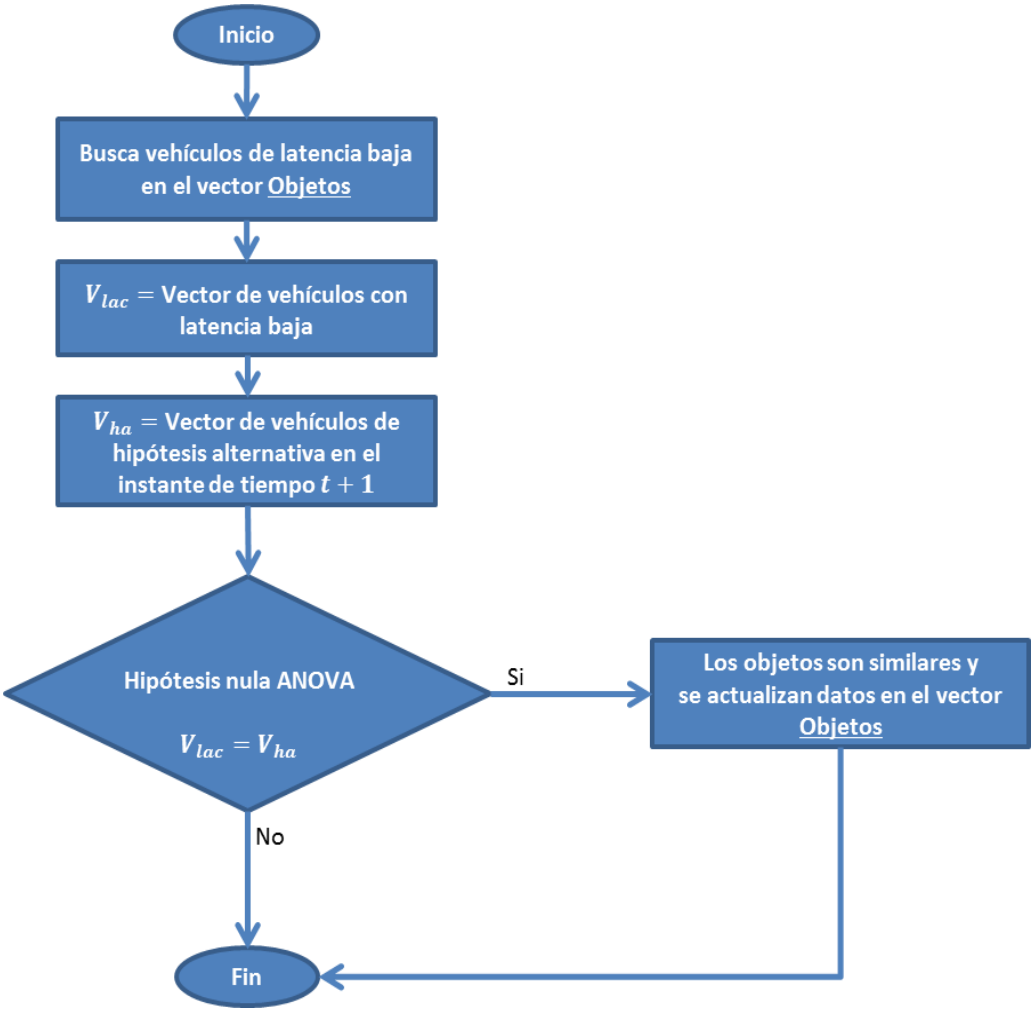


Figura 3-32 Algoritmo de latencia imperfecta.

- Función "**Nuevos vehículos**". Consiste en calcular el nivel de cercanía entre los objetos en el instante $t+1$ con hipótesis alternativa de ANOVA y las

fronteras determinadas a priori, a través de un umbral de distancias es posible determinar si un objeto acaba de entrar a escena:

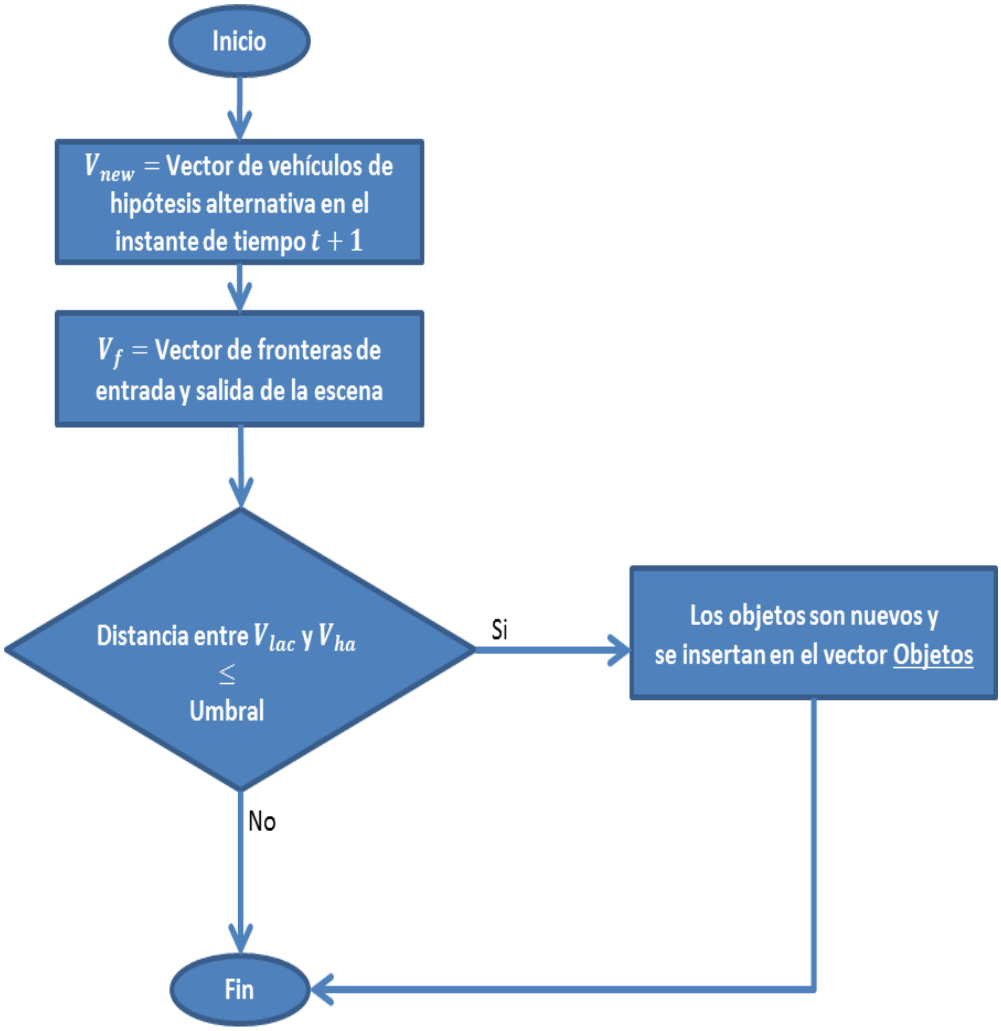
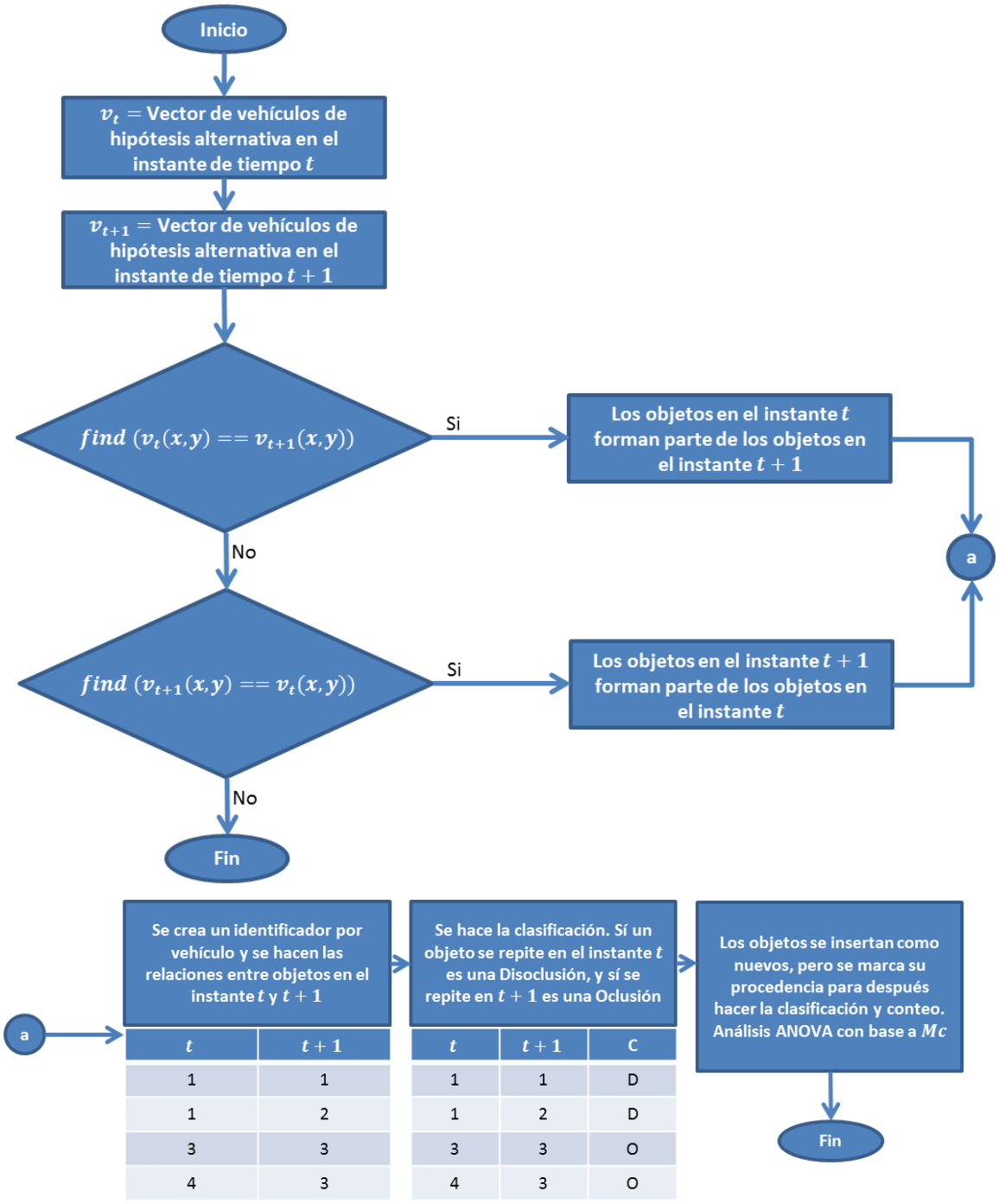


Figura 3-33 Algoritmo de vehículos nuevos.

- Función “**Oclusión/Disoclusión**”. Consiste en verificar si uno o más objetos en el instante t se han fusionado o separado con otro(s) en el instante $t + 1$:



t	$t+1$
1	1
1	2
3	3
4	3

t	$t+1$	C
1	1	D
1	2	D
3	3	O
4	3	O

Figura 3-34 Algoritmo de Oclusión/Disoclusión.

Es importante mencionar que los procesos descritos anteriormente trabajan en conjunto, el orden en que se ejecutan es esencial para su correcto funcionamiento. A continuación el orden de comunicación que existe entre dicho algoritmos y los parámetros que comparten:

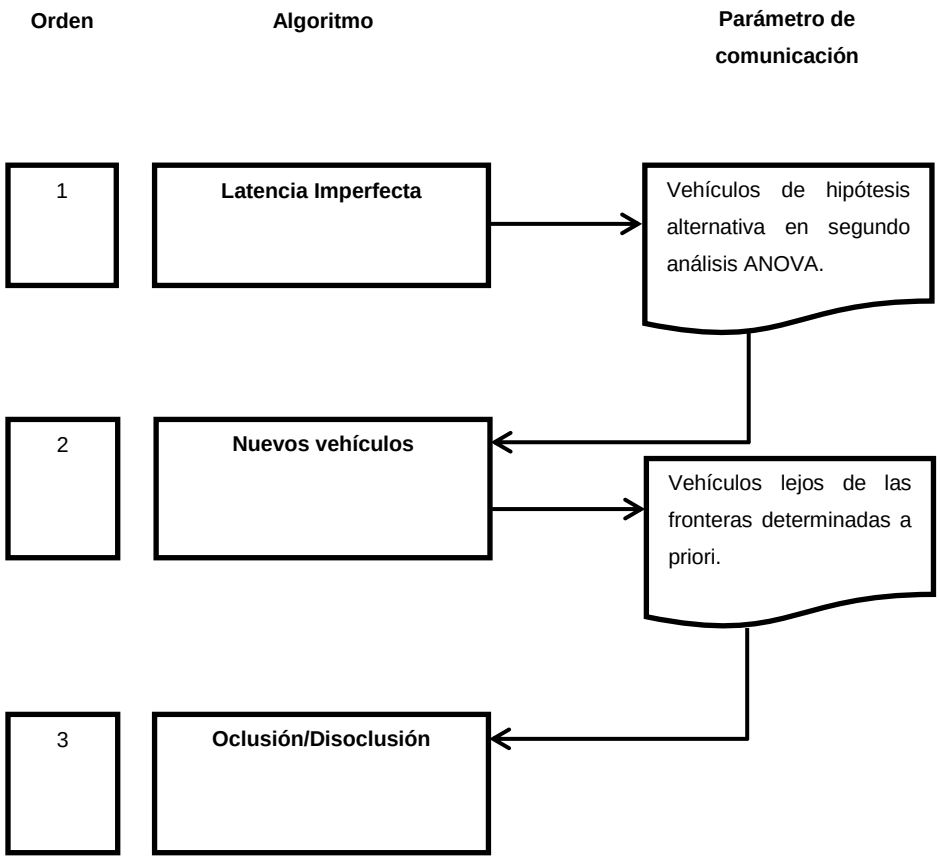
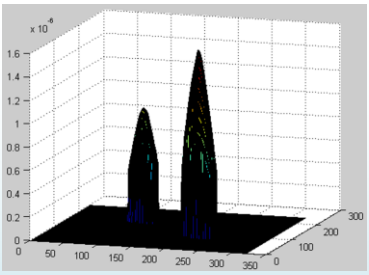


Figura 3-35 Comunicación entre los procedimientos de discriminación de vehículos no detectados.

3.8 Clasificación vehicular.

La clasificación vehicular dependerá del tamaño de ventana en la FDP propuesta y el nivel de reconocimiento que se desee, el sistema propuesto en esta tesis provee una herramienta que permite hacer una clasificación básica a través de la obtención de la altura de cada uno de los picos gaussianos, lo cual significa que entre más alto sea un pico, el objeto es más grande, a continuación un ejemplo:

a) Imagen original.



b) Surf FDP.

```
ml =
  1.0e-005 *
    0.1587    0.0969
>> mlx
mlx =
    96   162
>> mly
mly =
   214   109
```

c) Altura picos *gaussianos*.

donde:

ml	= Altura de los picos generados por la FDP.
mlx	= Posición abscisa en escena.
mly	= Posición ordenada en escena.

Figura 3-36 Ejemplo de los parámetros de clasificación con FDP.

Como se puede observar en observar en la **Figura 3-36**, la altura de los picos gaussianos generados por el procedimiento de *convolución Ventanas de Parzen* es una herramienta potencial para hacer una clasificación de objetos pequeños, medianos y grandes.

3.9 Modelo del sistema propuesto.

A continuación se presenta el esquema de procesos y herramientas que conforman la metodología propuesta en esta tesis:

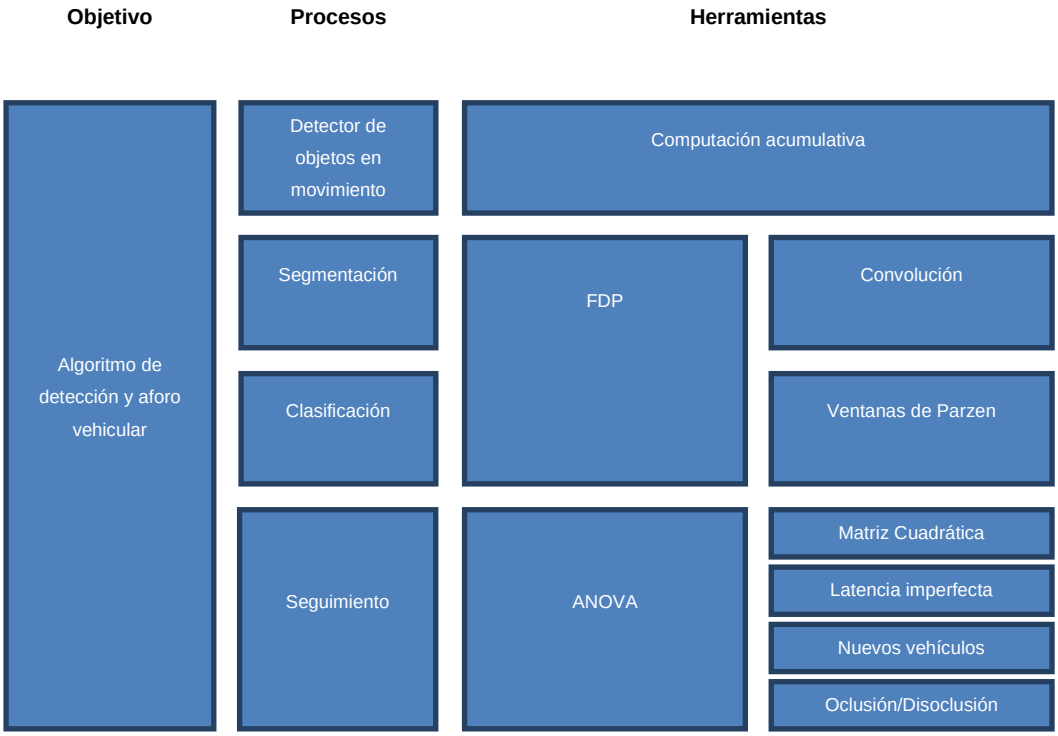


Figura 3-37 Esquema metodológico.

Es esencial plasmar de manera sistemática la metodología, una manera de representar la totalidad del sistema y la forma en la que se comportan y comunican cada uno de los procesos y herramientas, es con el siguiente modelo:

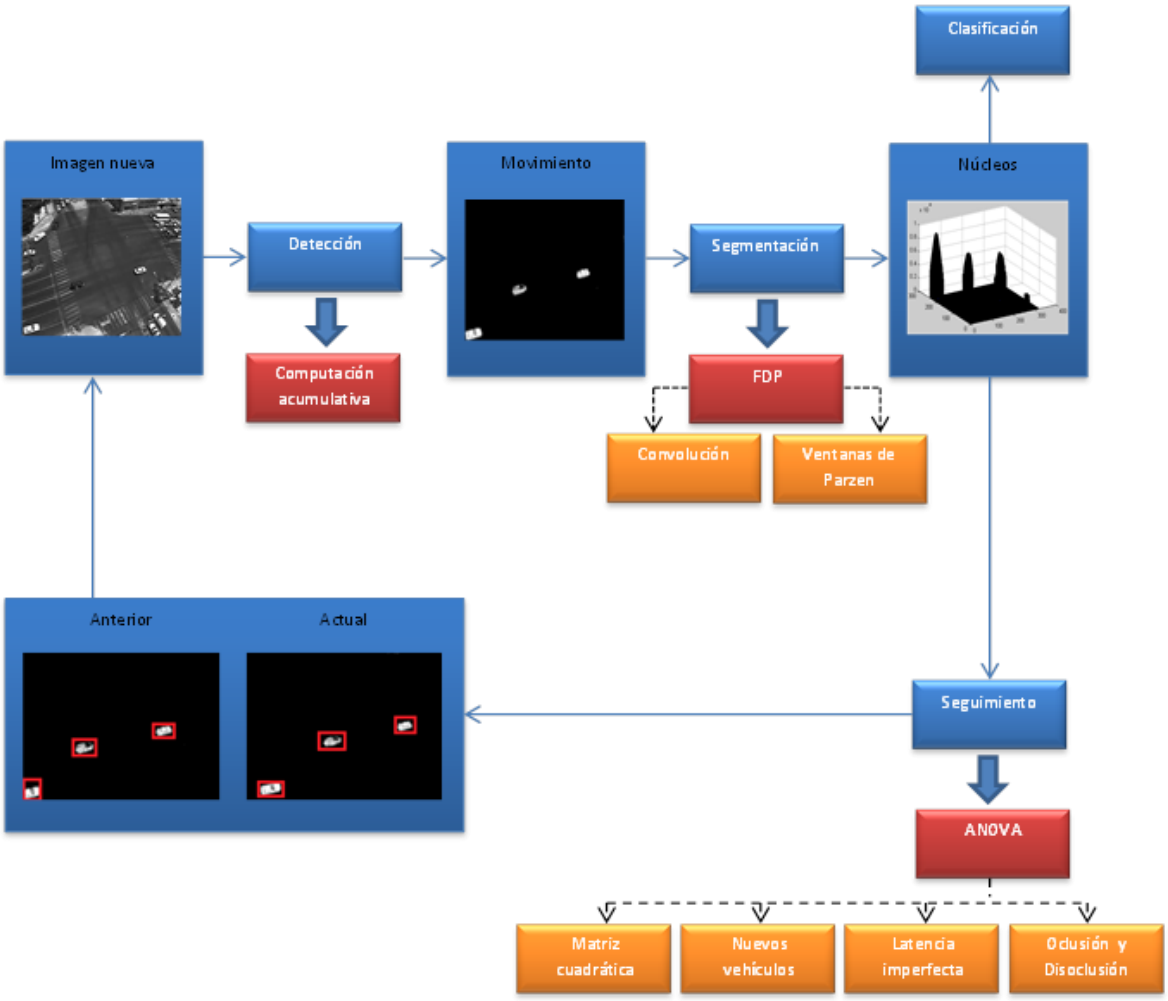


Figura 3-38 Modelo del Sistema.

4. RESULTADOS Y DISCUSIÓN.

El objetivo de este apartado es hacer una descripción de los resultados obtenidos en las técnicas propuestas en la tesis, expresar sus debilidades, fortalezas y el cumplimiento de los objetivos de la investigación planteada.

Detección.

Las imágenes son procesadas en escala de grises para mejorar el rendimiento de la aplicación y reducir los costos computacionales, a pesar de obtener dicha ventajas, es inevitable pasar por alto los efectos negativos, y es que al procesar imágenes en gris se corre el riesgo de que los objetos de poca reflectancia sean pasados por alto, esto sucede porque el nivel de intensidad entre los vehículos y la carretera puede ser muy parecido. La solución que se ha propuesto para determinar el umbral adecuado es extraer los dos vehículos de más baja reflectancia y que en la observación generan problemas para el proceso de computación acumulativa (tarda en reconocer que los objetos ya no son parte de un fondo), calcular su histograma (el cálculo se realizó ya habiendo restado el fondo y minimizando el ruido con una serie de filtros pasa bajos), obtener la desviación estándar y con base a ésta, seleccionar el mínimo valor de umbral. Este proceso también puede ayudar a determinar la operación o transformación que requieren las imágenes para ser procesadas y cumplir con el objetivo de tesis.

Se detectan dos colores de objeto (negro y gris) que son difíciles de ser eliminados por el proceso de computación acumulativa, esto es debido a que el nivel de reflectancia que emiten es muy baja, se cree que el nivel de umbral adecuado es el mínimo valor de dispersión calculado con base a los niveles de intensidad en los objetos. En el **Cuadro 4-1** se puede observar que los objetos negros son los más problemáticos, puesto que sus valores máximos de intensidad son casi la mitad de los elementos de color gris.

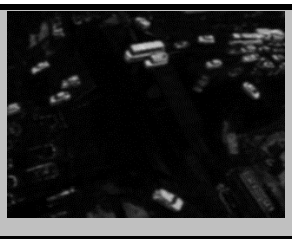
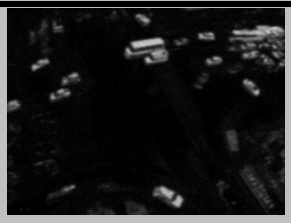
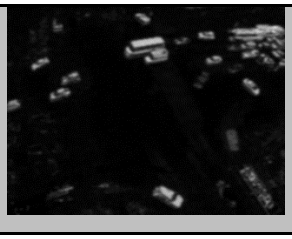
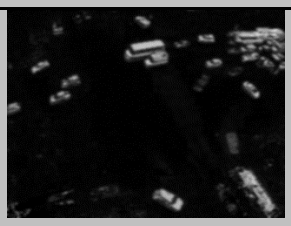
Cuadro 4-1 Objetos de baja reflectancia que maximizan el riesgo de ser pasados por alto utilizando la técnica de Computación Acumulativa.

Color	StdDev	Mean	Min	Max	Mode	MinVal
Negro	14.287	25.207	0	54	26(9)	10.92
Gris	28.141	40.585	0	112	22(11)	12.44

StdDev=Desviación estándar, Mean=Media, Min=Valor mínimo encontrado en la distribución, Max=Valor máximo encontrado en la distribución, Mode=moda, MinVal=Mínimo valor calculado.

En el **Cuadro 4-2** se puede observar que un valor de $U1 = 11$ es óptimo, ya que permite la generación de un fondo más adecuado, los valores por debajo de éste, permiten pasar una cantidad de ruido que incrementa conforme disminuye el valor de umbral (ver $U1 = 8 \text{ y } 5$), y los valores que están por arriba de él, discriminan una gran cantidad de píxeles que forman parte de objetos de baja reflectancia y que son de interés (ver $U1 = 15, 20 \text{ y } 30$).

Cuadro 4-2 Resultados obtenidos al aplicar distintos valores de $U1$ en el proceso de Computación Acumulativa.

Resultado	**Resultado**
	
$\boldsymbol{U1 = 30}$	$U1 = 20$
	
$\boldsymbol{U1 = 15}$	$U1 = 11$
	
$\boldsymbol{U1 = 8}$	$U1 = 5$

U1= Umbral.

El valor de permanencia depende de dos factores, la velocidad en la que los objetos se mueven y su tamaño, con esto se quiere decir, que si un vehículo circula lento y/o sus dimensiones son grandes, el sistema requiere ser comparado contra un valor de permanencia más alto. Se propone la observación de dos vehículos grandes a la hora de ejecutar dos de las acciones más lentas que permite la infraestructura del área de estudio.

El objetivo de la experimentación es encontrar un valor de U_{est} que elimine los espectros de los objetos, en vehículos grandes estos elementos pueden confundirse con los elementos en movimiento. El tiempo de los movimientos elegidos a observar (ver **Cuadro 4-3**) es de 18.15 segundos para el vehículo que da vuelta en U y 31.59 segundos para el que da vuelta a la derecha; dichos movimientos se han elegido porque el primero presenta el trayecto más complejo y que requiere mayor frenado, y el segundo porque ha presentado un congestionamiento considerable y su trayecto antes de desaparecer de la escena es de los más cortos, características que ayudan a reconocer los patrones de movimiento más extremos. Cabe señalar que dichos valores son tomados desde que el objeto aparece en la secuencia, hasta que desaparece de ella.

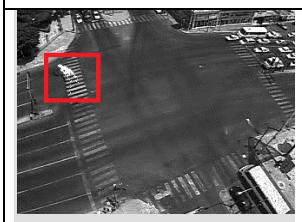
Podemos observar en el **Cuadro 4-3**, que el valor U_{est}= 90 es eficiente para el movimiento del vehículo 1, sin embargo en el vehículo 2 no es el adecuado, esto se debe a que el objeto 1 se mueve más rápido que el otro, por lo tanto el vehículo 2 requiere de un U_{est} mayor, en este caso con un valor de 130 logra superar el inconveniente.

Fue necesario identificar el valor mínimo de U_{est} a través de una serie de pruebas con el vehículo de velocidad más baja (con base a las muestras obtenidas se determina la velocidad mínima de $6.94\ \mathrm{km/hr}$), se determinó un valor de U_{est} mínimo de 120, sin embargo se dará una holgura de 10 valores más para vehículos con velocidades por debajo de lo determinado, por lo tanto U_{est} tendrá un valor constante de 130. Habiendo obtenido lo anterior es posible calcular U_{est} procediendo de forma análoga con la fórmula $U_{est} = \frac{(V_{min})(130)}{6.94}$, donde V_{min} es la velocidad mínima detectada en el área de estudio en kilómetros por hora.

Es importante mencionar que entre mayor sea el valor de U_{est} los objetos que se encontraban estáticos tardan más en eliminarse (verlo en el **Cuadro 4-3**), esto podría generar problemas a la hora de extraer los objetos en movimiento, puesto que los elementos que ya se han movido y no se han quitado del fondo serán producto de la resta de imágenes, sin embargo el problema se soluciona

con el algoritmo de detección de movimiento que también fue desarrollado en esta tesis.

Cuadro 4-3 Resultados obtenidos al aplicar distintos valores de permanencia U_{est} en el proceso de Computación Acumulativa.

Vehículo 1: Vuelta en U (Frame 4524)			
			
U_{est}: 50	U_{est}: 90	U_{est}: 130	U_{est}: 160
Vehículo 2: Vuelta a la derecha (Frame 3200).			
			
U_{est}: 50	U_{est}: 90	U_{est}: 130	U_{est}: 160

$U_{est} = Valor\ de\ permanencia$

Ya que el sistema toma como primer fondo el primer *frame* de la secuencia, es necesario determinar en qué momento la aplicación se estabiliza para ofrecer el fondo adecuado, por lo tanto fue necesario observar en qué número de *frame* elimina los objetos que se están moviendo, se propone correr seis videos diferentes y extraer el número de imagen en la que se estabiliza el algoritmo.

En el **Cuadro 4-4** se observa que la estabilidad depende de la primera imagen, entre más objetos en movimiento existan después del primer fondo, la aplicación tardará más en estabilizarse, sin embrago después de los 1000 *frames* (aproximadamente 46.88 segundos) en una secuencia con mucho tráfico, el sistema logra la estabilidad, se probó con las seis secuencias y resultó correcto.

Es posible proveer al sistema de un fondo inicial sin vehículos, esto haría que la aplicación logre una estabilidad casi inmediata, sin embargo se pretende automatizar al máximo este proceso y dejarle la responsabilidad al propio software.

Cuadro 4-4 Búsqueda del tiempo donde se genera un fondo adecuado para el proceso de Computación Acumulativa.

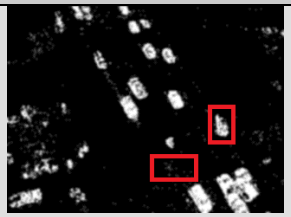
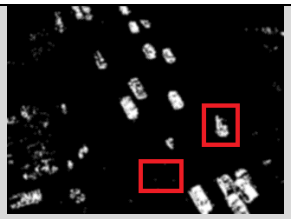
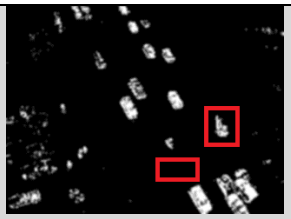
Secuencia	Número de frame	Tiempo (segundos)
Video 1	813	38.11
Video 2	370	17.35
Video 3	621	29.11
Video 4	492	23.06
Video 5	781	36.61
Video 6	461	21.61

Segmentación.

Es necesario usar un umbral U que ayude a perder el mínimo de píxeles en objetos de reflectancia parecida a la de la carretera sin dejar pasar aquellos que no pertenezcan a los elementos en movimiento, en el **Cuadro 4-5** se puede observar que el valor U = 30 es el más óptimo, puesto que con valores de 20 ó 25 produce mucho ruido sobre el espectro de los objetos en movimiento, y con valor 35 la eliminación de dicho ruido no es tan significativo al compararlo con el valor de 30.

Es importante mencionar que la elección del umbral depende de la perdida de píxeles en los elementos de interés, ya que al ir incrementando a U éstos se van perdiendo, por lo tanto se elige un valor medio que aunque no exacto puede superar el ruido a través de filtros de suavizado o erosiones.

Cuadro 4-5 Búsqueda de un umbral U óptimo para el proceso de Segmentación a la hora de hacer la resta absoluta entre el fondo los objetos de interés.

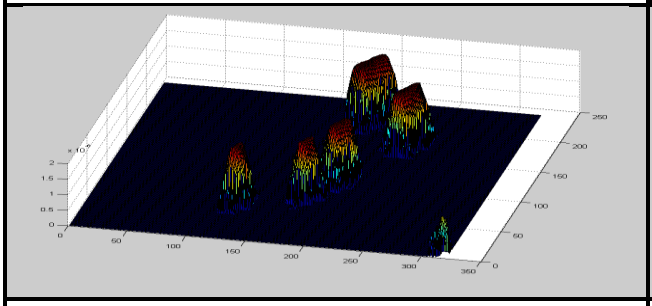
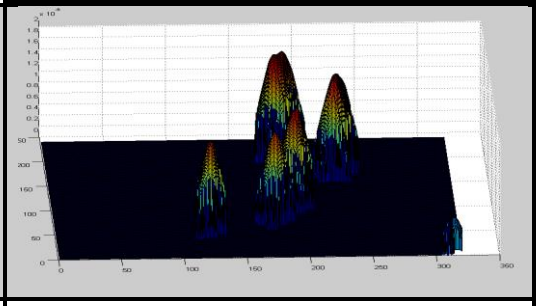
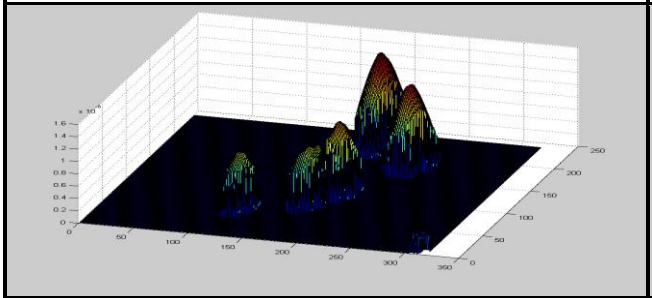
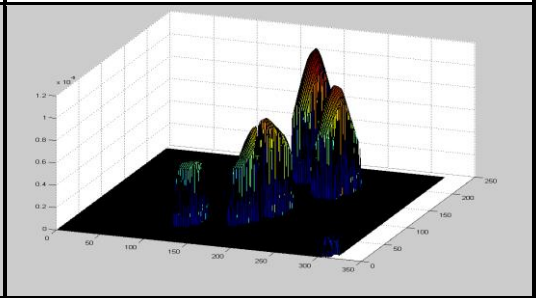
Resultado	Resultado
	
U > 20	U > 25
	
U > 30	U > 35

U=Umbral.

La eficiencia en el proceso de segmentación dependerá del tamaño de la *Ventana de Parzen* que se aplique a cada uno de los *frames*, es ésta la que delimita la cantidad de píxeles que formarán parte del núcleo de cada objeto detectado.

En el **Cuadro 4-6** se puede observar que $\mathrm{h} = 10$ es un tamaño de ventana óptimo, ya que los picos gaussianos de cada objeto están bien identificados, con $\mathrm{h} = 5$ se comienza a generar más de un pico por objeto y con $\mathrm{h} = 15$ ó 20 se comienzan a oclusionar.

Cuadro 4-6 Búsqueda de un tamaño de Ventana de Parzen óptimo para el proceso de Segmentación.

Frame de muestra.	
	
$h = 5$	$h = 10$
	
$h = 15$	$h = 20$

h = Tamaño de la *Ventana de Parzen*.

Seguimiento.

El proceso de seguimiento se puede resumir en la detección de cada uno de los picos de los objetos para ser comparados a través de un análisis de varianza ANOVA entre cada *frame*.

Como se puede observar en el **Cuadro 4-7**, un valor de nivel de significancia $\propto = 0.02$ es óptimo, ya que ha detectado que en un instante de tiempo un objeto no se reconoce (Ha), sin embargo el proceso de Latencia Imperfecta logra encontrarlo en el historial de vehículos desaparecidos (Hn), logrando la detección y seguimiento de los tres vehículos en la escena.

A pesar de que casi todos los valores de significancia en la experimentación del **Cuadro 4-7** excepto $\propto = 0.06$ logran seguir a los tres vehículos en escena, el nivel de error (Error) y los instantes de tiempo donde se cumple la hipótesis alternativa (Ha) incrementa, lo cual quiere decir que existen más instantes de tiempo donde los objetos no se han reconocido.

Cuadro 4-7 Búsqueda de un nivel de significancia $\propto$ óptimo para el proceso de Seguimiento en el análisis de varianza.

Frames	Veh	$\propto$	Ha	New	L_imp	Hn	Tv	Error
36	3	0.01	2	2	2	1	3	6%
36	3	0.02	1	1	2	1	3	3%
36	3	0.03	3	1	2	1	3	8%
36	3	0.04	3	1	2	1	3	8%
36	3	0.05	3	1	2	1	3	8%
36	3	0.06	5	2	3	2	4	14%

Frames=total de *frames* en la secuencia de estudio, Veh=Total de vehículos aparecidos en la secuencia, $\propto$=Nivel de significancia para el análisis de varianza, Ha=Instantes donde la hipótesis alternativa se cumple, New=Detección de vehículos nuevos, L_imp=Número de veces donde el algoritmo encuentra en el historial vehículos de latencia imperfecta, Hn= Instantes donde la hipótesis nula se cumple, Tv=Total de vehículos registrados al final de la secuencia, Error=Nivel de error.

En el **Cuadro 4-8** se puede observar que al incrementar el número de vehículos en escena el nivel de error incrementa, esto es debido principalmente a que el número de oclusiones tiende a incrementar, sin embargo, otros factores pueden ser:

- Objetos de baja reflectancia y que se van alejando del sensor de video, lo cual puede afectar al proceso de análisis de varianza ANOVA, el cual tiene como principal objetivo validar la similitud entre los vehículos de *frame* a *frame*.
- El tamaño de la ventana en la estimación de la función de densidad, ya que entre más grande sea, la posibilidad de oclusiones incrementa, por otro lado, si es muy pequeña el número de disoclusiones aumenta.
- Vehículos que al comenzar en escena están muy cerca de las fronteras de salida detectadas, por lo tanto serán discriminados por el algoritmo de detección.

Cuadro 4-8 Evaluación del conjunto de algoritmos propuestos.

Num_m	N_veh_ini	N_veh_fin	D_ini	D_fin	O_ini	O_fin	D_ocl	D_dis	Dis	Error
1	6	6	3	4	2(2,3)	2(2,3)	0	1	1	33%
2	2	3	2	3	0	0	0	0	0	0%
3	6	5	5	4	1(1)	0	1(1)	1(1)	1	17%
4	7	9	7	7	0	0	2(1,1)	2(1,1)	0	22%
5	9	11	12	17	0	5	5(1,2,3,2,1)	0	0	54%

Num_m=Número de la muestra a evaluar, N_veh_ini=Número de vehículos al inicio de la secuencia, N_veh_fin=Número de vehículos al final de la secuencia, D_ini=Número de vehículos detectados por el algoritmo al inicio de la secuencia, D_fin=Número de vehículos detectados por el algoritmo al final de la secuencia, O_ini=Número de objetos ocluidos al inicio de la secuenciam, O_fin=Número de objetos ocluidos al final de la secuencia, D_ocl=Número de oclusiones detectados por el algoritmo, D_dis=Número de disoclusiones detectados por el algoritmo, Dis=Disoclusiones en la secuencia, Error=Nivel de error.

5. CONCLUSIONES Y TRABAJOS FUTUROS.

A continuación se presentan las conclusiones obtenidas después de haber generado los resultados de la experimentación y algunas propuestas para trabajos a futuro:

- Es posible mejorar el rendimiento del sistema de aforo vehicular automático mediante el uso de técnicas de programación en paralelo, cabe mencionar que la mayoría de las funciones desarrolladas en esta tesis pueden paralelizarse a través de hilos, ya que sus procedimientos son independientes y no requieren comunicación entre ellos. CUDA (Arquitectura Unificada de Dispositivos de Cómputo) podría ser una gran oportunidad, ya que este compilador tiene la capacidad de explotar los núcleos del CPU y de forma independiente su Unidad de Procesamiento gráfico.
- Una aplicación de visión artificial robusta requiere la capacidad de controlar las oclusiones/disoclusiones, en esta tesis este factor depende altamente de una calibración de los umbrales de bajo nivel de error y de la correcta identificación de las fronteras.
- Aunque en esta tesis no se tratan aquellos objetos que de principio a fin de la escena están ocluidos, se observa que se pueden inferir tomando un umbral de tamaño de referencia, y así obtener un registro correcto de ellos.
- En la parte experimental surgieron algunos inconvenientes por utilizar una sola cámara, por la distorsión que afectaba a los objetos cuando se alejaban de su área de visión, por lo tanto se propone el uso de un grupo de cámaras calibradas en altura y distancia, alimentando un sistema integral.
- Se propone como trabajo futuro y para la técnica de seguimiento en esta tesis, desechar cualquier objeto detectado lejos de las fronteras determinadas y sólo seguir aquellos que se mueven muy cerca y entre sí, así como lo hace el filtro de partículas. Con esto se logra discriminar con mayor eficiencia el ruido causado por los factores de luminosidad y e incrementar el rendimiento a más de un 50%, de forma tal que se buscarán los núcleos en el instante actual (t) con base a una búsqueda delimitada por un umbral de distancia entre los núcleos detectados en el instante anterior (t-1).

- Para investigaciones futuras, una buena oportunidad sería hacer un análisis sobre la clasificación vehicular a través del cálculo de la altura de los picos gaussianos, la experimentación se podría realizar con base a la desviación estándar entre las muestras de un conjunto de vehículos de distintos tamaños, de esta manera es posible proponer umbrales de tamaño de objetos para determinar una clasificación más robusta.
- La calibración del sistema propuesto en esta tesis debe tomar en cuenta que los cambios drásticos en el tamaño de la *Ventana de Parzen* afectará la clasificación vehicular y al proceso de segmentación. Esto significa que una ventana grande facilita la clasificación pero incrementa la probabilidad de que existan más fusiones de objetos (oclusiones), por otro lado, una ventana pequeña disminuye la existencia de oclusiones pero σ en la altura de los picos gaussianos será baja, evento que hará que las alturas en los picos sean muy parecidas y esto generará algunas dificultades a la hora de clasificar.

6. REFERENCIAS BIBLIOGRAFÍACAS.

Egües, G. R. 1964. *Manual de Ingeniería de Tránsito*. Chicago, Illinois: Reuben H. Donnelly Corporation.

Larry, J. G. 1990. *Cálculo y sus Aplicaciones*. (4ta. Ed.). México: Prentice Hall.

Delgado, S. J. 1992. *Algunos problemas básicos del Análisis de Varianza.* (Ed.). Salamanca, España: Ediciones Universidad de Salamanca.

Walpole, R. E. 1999. *Probabilidad y Estadística para Ingenieros.* (6Ta. Ed.). México: Prentice Hall.

Hilario, J. M., Collado, J. M. y Armingol, A. (2000). *Detección de peatones para vehículos inteligentes basada en modelos de contornos activos y visión estéreo.* Obtenido el 20 de noviembre de 2012, desde el Grupo de Sistemas Inteligentes – Escuela Politécnica Superior: http://www.aurova.ua.es:8080/ja2005/comu/3281-JMA-JA2005.pdf

Fernández, C. M., García, M. M., Alonso, O. G., Cano, R. J. y Solares, S. J. 2000. *Técnicas para el mantenimiento y diagnóstico de máquinas eléctricas rotativas*. (Ed.). España: Marcombo Boixareu Editores.

Díaz, P. y Fernández, P. (2001). *Métodos no paramétricos para la comparación de dos medias.* Obtenido el 9 de diciembre de 2012, desde la Unidad de epidemiología Clínica y Bioestadística – Complexo Hospitalario Juan Canalejo de Coruña: http://www.fisterra.com/mbe/investiga/t_student/t_student2.pdf

Blaiotta, J. y Delieutraz, P. (2004). *MTeorema Central del Límita.* Obtenido el 12 de diciembre de 2012, desde la Facultad de Ciencias Exactas y Naturales de la Universidad de Buenos Aires: http://www.academia.edu/3322286/La_demostracion_del_Teorema_Central_del_Limite.

Crisóstomo, P. y Yoshimoto. (2005). *Arquitectura para el procesamiento de imágenes de tránsito vehicular*. Obtenido el 12 de noviembre de 2012, desde el Grupo de Procesamiento Digital de Señales e Imágenes - Pontificia Universidad Católica del Perú: http://tutr2.com/view/196547

Servy, E., Cuesta, C. y Gonzalo, M. (2005). *Estimación de funciones de densidad de datos correlacionados y/o con valores atípicos.* Obtenido el 20 de noviembre de 2012, desde Decimas Jornadas Investigación en la facultad de ciencias económicas y estadísticas de la Universidad Nacional del Rosario: http://www.fcecon.unr.edu.ar/investigacion/jornadas/archivos/mariycuesta.pdf

Yilmaz, A., Javed, O. y Shah, M. (Diciembre, 2006). Object Tracking: A Survey. *ACM Computing Surveys,* 38(4).

Zambrano, G. M., Parra, C. A., Manrique, M. R. y Bustacara, C.J. (Junio, 2007). Estación de control de calidad por visión artificial para un centro de manufactura integrada por computador (CMI). *Ingeniería y Universidad*, 11(1). pp. 33-55.

Maldonado, M., Gallegos, R., Sandoval, J.A., López, F. y Cabrera, M. (Octubre, 2007). Hacia un Sistema Automático de Caracterización Vehicular. *Congreso Nacional de Control Automático A.M.A.C.* pp. 1-6. México: Instituto de ingeniería Civil-UANL.

Rueda, D.L., Narváez, L.R. y Sotaquirá, G.M. (Septiembre, 2008). Algoritmo de Apoyo para el Diagnóstico de Tuberculosis Pulmonar Mediante el Procesamiento Digital de Imágenes. *XIII Simposio de tratamiento de señales, imágenes y visión artificial.* pp. 27-30. Colombia: Universidad Santo Tomas de Bucaramanga. ISSN: 978-958-8477-00-8.

Huang, C., Wu, Bo. y Nevatia, R. (Octubre, 2008). Robust Object Tracking by Hierarchical Association of Detection Responses. *10th European Conference on Computer Vision.* pp. 788-801. Francia: University of Southern California. ISBN: 978-3-540-88685-3.

Nope, S. E., Loaiza, H. y Caicedo, E. (Diciembre, 2008). Estudio comparativo de técnicas para el reconocimiento de gestos por visión artificial. *Avances en Sistemas e Informática*, 5(3). pp. 1657-1663.

Ferraz, A. y Lameda, C. (Diciembre, 2008). Extracción automática de características de vehículos en movimiento a partir de videos basada en red neuronal de Kohonen. *INGENIERÍA UC*, 15(3). pp. 33-44.

Faytong, J. E. y Moggia, G. J. (2008). *Monitoreo automático de carreteras mediante el uso de un sistema de detección, seguimiento y extracción de características de vehículos con técnicas de visión por computador.* Tesis de ingeniería. Escuela Superior Politécnica del Litoral. Ecuador.

Pajares, G. y de la Cruz, J.M. 2008. *Visión por Computador. Imágenes digitales y aplicaciones*. (2da. Ed.). Madrid, España: RA-MA Editorial.

Garrido, J. J. (2008). *Seguimiento de personas y vehículos por videovigilancia inteligente.* Tesis de Ingeniería. Universidad Autónoma de Barcelona. Bellaterra, Barcelona.

Gómez, E. (2009). *Aplicación del filtro de partículas al seguimiento de un objeto en 2D y extracción de características.* Tesis de Ingeniería. Universidad Rey Juan Carlos, Madrid, España.

García, L. P. y Sancho, G. L. (Mayo 2010). Estimación de la densidad de probabilidad mediante ventanas de Parzen. *III Jornadas de Introducción a la investigación de la UPCT.* pp. 68-70. Colombia: Universidad Politécnica de Cartagena. ISSN: 1888-8356.

Leal, N., Leal, E. y Willian, J. (Diciembre, 2010). Sistemas de monitoreo de tránsito vehicular basados en técnicas de segmentación de imágenes*. Avances en Sistemas e Informática,* 7(3). pp. 75-86.

Gutiérrez, G. J. (2010). *Filtro de partículas para Seguimiento de grupos*. Proyecto fin de carrera. Universidad Rey Juan Carlos, Madrid, España.

Juric, K. S., Marcovic, I. y Petrovic, I. (2010). *People Traking with Heterogeneous Sensors using JPDAF with Entropy Based Track Managemen*. Obtenido el 20 de noviembre de 2012, desde el Department of Control and Computer Engineering – Faculty of Electrical Engineering and Computing: https://bib.irb.hr/datoteka/519441.MSMT_Tracking_ECMR2011.pdf

Garzón, G. E. (2010). *Análisis de Imágenes: Sistema de Seguimiento de Personas.* Proyecto fin de carrera. Universidad Pontificia de Comillas, Madrid, España.

Levin, R. y Rubin, D. 2010. *Estadística para Administración y Economía.* (7ma. Ed.). México: Prentice Hall.

Iglesis, M. P. (2010). *Estrategias de clasificación de texturas en imágenes forestales hemisféricas.* Tesis de Maestría. Universidad Complutense de Madrid. España.

Álvarez, T. J. (2010). *Ensamblador Versus C en Microcontroladores.* Obtenido el 12 de marzo de 2014, desde TECNOLOGÍA EN BOLIVIA: http://www.tecbolivia.com/index.php/articulos-y-tutorialesmicrocontroladores/6-ensamblador-versus-c-en-microcontroladores

Valverde, R. J. (2011). *Seguimiento Asistido de Objetos en Video utilizando Contornos Activos y Conjuntos de Nivel.* Obtenido el 20 de noviembre de 2012, desde las Actas de las Jornadas Chilenas de Computación: http://intranet.usat.edu.pe/usat/idsistemas/2011/08/17/jornadas-chilenas-de-computacion-2011/

Márquez, F. J. (2011). Curso procesamiento y análisis de imágenes. *Carteles CCADET 2011.* México: Universidad Nacional Autónoma de México. ISSN: 1888-8356.

Martín, I. N. (2011). *Un estudio basado en la Técnica de Mean Shift para Agrupamiento y Seguimiento en video.* Tesis de Licenciatura. Universidad de Buenos Aires. Argentina.

Rodríguez, M. R. y Sossa, A, H. 2012. *Procesamiento y Análisis Digital de Imágenes*. (Ed.). Madrid, España: RA-MA Editorial.

Vélez, J. S. (2013). *Lenguajes para Visión Artificial.* Obtenido el 12 de marzo de 2014, desde Noticias, curiosidades, artículos de opinión y tutorial sobre Visión por Computador: http://visionporcomputador.es/2011_04_01_archive.html

Dirección General de Servicios Técnicos (2014). Datos Viales 2014. *Subsecretaría de Infraestructura.* México: Secretaría de Comunicaciones y Transporte. http://www.sct.gob.mx/fileadmin/DireccionesGrales/DGST/Datos-Viales-2014/INTRODUCCI%C3%93N.pdf

Anexo 1. Tabla de distribución normal de valores positivos.

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
0.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
0.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
0.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
0.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
0.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
0.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
0.7	.7580	.7611	.7642	.7673	.7704	.7734	.7764	.7794	.7823	.7852
0.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
0.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9429	.9441
1.6	.9452	.9463	.9474	.9484	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9699	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9761	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9875	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9987	.9987	.9988	.9988	.9989	.9989	.9989	.9990	.9990
3.1	.9990	.9991	.9991	.9991	.9992	.9992	.9992	.9992	.9993	.9993
3.2	.9993	.9993	.9994	.9994	.9994	.9994	.9994	.9995	.9995	.9995
3.3	.9995	.9995	.9995	.9996	.9996	.9996	.9996	.9996	.9996	.9997
3.4	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9998

Anexo 2. Tabla de distribución normal de valores negativos.

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
−3.4	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0002
−3.3	.0005	.0005	.0005	.0004	.0004	.0004	.0004	.0004	.0004	.0003
−3.2	.0007	.0007	.0006	.0006	.0006	.0006	.0006	.0005	.0005	.0005
−3.1	.0010	.0009	.0009	.0009	.0008	.0008	.0008	.0008	.0007	.0007
−3.0	.0013	.0013	.0013	.0012	.0012	.0011	.0011	.0011	.0010	.0010
−2.9	.0019	.0018	.0018	.0017	.0016	.0016	.0015	.0015	.0014	.0014
−2.8	.0026	.0025	.0024	.0023	.0023	.0022	.0021	.0021	.0020	.0019
−2.7	.0035	.0034	.0033	.0032	.0031	.0030	.0029	.0028	.0027	.0026
−2.6	.0047	.0045	.0044	.0043	.0041	.0040	.0039	.0038	.0037	.0036
−2.5	.0062	.0060	.0059	.0057	.0055	.0054	.0052	.0051	.0049	.0048
−2.4	.0082	.0080	.0078	.0075	.0073	.0071	.0069	.0068	.0066	.0064
−2.3	.0107	.0104	.0102	.0099	.0096	.0094	.0091	.0089	.0087	.0084
−2.2	.0139	.0136	.0132	.0129	.0125	.0122	.0119	.0116	.0113	.0110
−2.1	.0179	.0174	.0170	.0166	.0162	.0158	.0154	.0150	.0146	.0143
−2.0	.0228	.0222	.0217	.0212	.0207	.0202	.0197	.0192	.0188	.0183
−1.9	.0287	.0281	.0274	.0268	.0262	.0256	.0250	.0244	.0239	.0233
−1.8	.0359	.0351	.0344	.0336	.0329	.0322	.0314	.0307	.0301	.0294
−1.7	.0446	.0436	.0427	.0418	.0409	.0401	.0392	.0384	.0375	.0367
−1.6	.0548	.0537	.0526	.0516	.0505	.0495	.0485	.0475	.0465	.0455
−1.5	.0668	.0655	.0643	.0630	.0618	.0606	.0594	.0582	.0571	.0559
−1.4	.0808	.0793	.0778	.0764	.0749	.0735	.0721	.0708	.0694	.0681
−1.3	.0968	.0951	.0934	.0918	.0901	.0885	.0869	.0853	.0838	.0823
−1.2	.1151	.1131	.1112	.1093	.1075	.1056	.1038	.1020	.1003	.0985
−1.1	.1357	.1335	.1314	.1292	.1271	.1251	.1230	.1210	.1190	.1170
−1.0	.1587	.1562	.1539	.1515	.1492	.1469	.1446	.1423	.1401	.1379
−0.9	.1841	.1814	.1788	.1762	.1736	.1711	.1685	.1660	.1635	.1611
−0.8	.2119	.2090	.2061	.2033	.2005	.1977	.1949	.1922	.1894	.1867
−0.7	.2420	.2389	.2358	.2327	.2296	.2266	.2236	.2206	.2177	.2148
−0.6	.2743	.2709	.2676	.2643	.2611	.2578	.2546	.2514	.2483	.2451
−0.5	.3085	.3050	.3015	.2981	.2946	.2912	.2877	.2843	.2810	.2776
−0.4	.3446	.3409	.3372	.3336	.3300	.3264	.3228	.3192	.3156	.3121
−0.3	.3821	.3783	.3745	.3707	.3669	.3632	.3594	.3557	.3520	.3483
−0.2	.4207	.4168	.4129	.4090	.4052	.4013	.3974	.3936	.3897	.3859
−0.1	.4602	.4562	.4522	.4483	.4443	.4404	.4364	.4325	.4286	.4247
−0.0	.5000	.4960	.4920	.4880	.4840	.4801	.4761	.4721	.4681	.4641

Anexo 3. Algoritmos importantes.

Calculo de la FDP.

Código Matlab para generar la Función de Densidad de Probabilidad de *Ventanas de Parzen* de una imagen:

```
function pd = CalculoConvolucion(rp,cp,vren,vcol,h,N)
%   rp    = Posición renglón
%   cp    = Posición columna
%   vren  = Vector renglón conjunto
%   vcol  = Vector columna conjunto
%   h     = Ancho de kernel
%   N     = Tamaño de muestra
inc=1;
pd=0;
    while inc <= length(vren)
    xi=vren(inc);
    yi=vcol(inc);
    pd=pd+(1/((2*h+1)^2))*exp(-((rp-xi)^2+(cp-yi)^2)/(2*((2*h+1)^2)));
    inc=inc+1;
    end
    pd=1/(2*pi*N)*pd;
end
```

Análisis de varianza ANOVA.

Código Matlab que busca la similitud de varianza entre los objetos de un *frame* a otro con base a un análisis de varianza ANOVA:

```
function [encuentrar,encuentrac] =
Valida_similitud(m,img1,img2,img1x,img1y,img2x,img2y)
%   m           = matriz de distancias
%   img1        = Id de la imagen 1
%   img2        = Id de la imagen 2
%   img1x       = vector posiciones en x de la imagen 1
%   img1y       = vector posiciones en y de la imagen 1
%   img2x       = vector posiciones en x de la imagen 2
%   img2y       = vector posiciones en y de la imagen 2
[r,c]=size(m);
ren=1;
simi=1;
 encuentrar=zeros();
 encuentrac=zeros();
while(ren<=r)
    col=2;
    ban=0;
    while(col<c && ban==0)
    v_muestra=zeros();
    v_muestra2=zeros();
    v_union=zeros();

    position=m(ren,1);
    px=img1x(position);
    py=img1y(position);
    [vr,vc] = MatrizConvolucion(px,py,240,320,img1,3);
    v_muestra = Crea_muestra(vr,vc,img1);

    position=m(ren,col);
    px2=img2x(position);
    py2=img2y(position);
    [vr2,vc2] = MatrizConvolucion(px2,py2,240,320,img2,3);
    v_muestra2 = Crea_muestra(vr2,vc2,img2);

    v_union= Union(v_muestra,v_muestra2);
    [sst,sse,k,nk,mstr,mse,f,pc]=Similitud(v_union,0.94);%0.98 (óptimo)
    fprintf(num2str(f));
    fprintf('-');
    fprintf(num2str(pc));
    if(f < pc)
        ban=1;
        encuentrar(simi)=ren;
        encuentrac(simi)=col;
        simi=simi+1;
    end
    col=col+1;
    end
ren=ren+1;
end
end
```

Oclusión y Disoclusión.

Código Matlab que genera una tabla donde se identifican los vehículos oclusionados y disoclusionados, además de sus procedencias:

```
function [vne,moc] =
Oclu_Diso_con_MatrizConvolucion(mc,vtx,pva,mlx,mly,c_t_vx,c_t_vy)
% Verifica si hubo alguna oclusión o disoclusión.
%VARIABLES DE ENTRADA:
%   mc      = Matriz cuadratica de cercania.
%   vtx     = Vector posición en x hipotesis alternativa en el instante t
(en mc)
%   mlx     = Vector posiciones en x en el instante t+1.
%   mly     = Vector posiciones en y en el instante t+1.
%   c_t_vx  = Vector posiciones en x en el instante t.
%   c_t_vy  = Vector posiciones en y en el instante t.
%   pva     = Posiciones en el instante t+1 con hipotesis alternativa
lejos de fronteras de entrada (en mlx,mly)
%VARIABLES DE SALIDA:
%   vne  = Vehiculos no encontrados
%   moc  = Matriz clasificada Olusión/Disoclusión
d=42; %Umbral de distancia (el verdadero valor es 15, se modifica para
pruebas)
[r1,c1]=size(pva);
[r2,c2]=size(vtx);
inc1=1;
m_oclusion=(zeros);
m_ocl_dis=(zeros);
inc_oc=1;
h=20;%tamñao de ventana (Parzen)
tam_r=240;%tamaño de imagen en renglones
tam_c=320;%tamaño de imagen en columnas
while(inc1 <= c1)
    p1x=mlx(pva(inc1));
    p1y=mly(pva(inc1));
    inc2=1;
    while(inc2 <= c2)
    pos=mc(vtx(inc2),1);
    p2x=c_t_vx(pos);
    p2y=c_t_vy(pos);

    [vro,vco] = MatrizConvolucion_con_vacios(p1x,p1y,tam_r,tam_c,h);
    [vrd,vcd] = MatrizConvolucion_con_vacios(p2x,p2y,tam_r,tam_c,h);

    if(length(find(vro==p2x)) &&  length(find(vco==p2y)))
            m_oclusion(inc_oc,1)=pva(inc1);
            m_oclusion(inc_oc,2)=pos;
            inc_oc=inc_oc+1;
    else
        if(length(find(vrd==p1x)) &&  length(find(vcd==p1y)))
            m_oclusion(inc_oc,1)=pva(inc1);
            m_oclusion(inc_oc,2)=pos;
            inc_oc=inc_oc+1;
        end
    end
```

```
    inc2=inc2+1;
    end
inc1=inc1+1;
end
%Verificar si es oclusión ó disoclusión y poner una marca en la sig.
%columna de la matriz.
[rod,cod]=size(m_oclusion);
if(rod > 1)%si la matriz no tiene más de un renglón significa que hay un
           %sólo objeto y que es muy posible que por el nivel de
           %luminosidad no haya sido detectado por ANOVA
m_ocl_dis = Clasifica(mc,m_oclusion);
end
vne= m_oclusion;
moc=m_ocl_dis;
end
```